IBM

IBM 系統 - iSeries

程式設計
IBM Toolbox for Java

版本 5 版次 4

IBM

IBM 系統 - iSeries
程式設計
IBM Toolbox for Java

版本 5 版次 4

請注意

使用此資訊及其支援的產品之前，請先閱讀第 703 頁的『注意事項』中的資訊。

第十版 (2006 年 2 月)

此版本適用於 IBM Toolbox for Java (產品編號 5722-JC1) 版本 5 版次 4 修正層次 0，以及所有後續的版次和修訂版 (除非新版中另有指示)。此版本並非適用於所有的精簡指令集電腦 (RISC) 機型和 CISC 機型。

目錄

IBM Toolbox for Java

IBM® Toolbox for Java™ 爲一組 Java 類別，可讓您使用 Java 程式來存取您 iSeries™ 伺服器上的資料。您可以使用這些類別，來撰寫主從架構應用程式、Applet 及 Servlet，以便與 iSeries 中的資料搭配使用。您也可以在 iSeries Java 虛擬機器 (JVM) 上，執行使用 IBM Toolbox for Java 類別的 Java 應用程式。

IBM Toolbox for Java 使用 iSeries 主電腦伺服器作爲對系統的存取點。因爲 IBM Toolbox for Java 使用內建於 Java 的通訊功能，所以您不必透過 IBM iSeries Access for Windows®，來使用 IBM Toolbox for Java。每一個伺服器均是以伺服器的個別工作之方式執行，而每一個伺服器工作會在 socket 連接中傳送與接收資料串流。

註: 使用程式碼範例，即表示您同意第 700 頁的『程式碼授權及免責聲明資訊』的條款。

新增功能

本主題的重點在於對 V5R4 中的 IBM Toolbox for Java 進行的變更。

IBM Toolbox for Java 可以下列方式提供：

- IBM Toolbox for Java 5722-JC1 版本 5 版次 4 (V5R4) 的授權程式，可安裝在 i5/OS™ V5R2 與以上的版本上。IBM Toolbox for Java 可以從用戶端連接回 i5/OS V5R2 與以上的版本。
- i5/OS 也包含非圖形式的 IBM Toolbox for Java 類別，這些類別已經過最佳化，可在 iSeries Java 虛擬機器 (JVM) 上執行 IBM Toolbox for Java 類別時使用。因此，假如您不需要使用此授權程式的圖形功能時，您仍然可以輕鬆地使用 IBM Toolbox for Java。如需詳細資訊， 請參閱 Jar 檔案。
- IBM Toolbox for Java 也提供開放原始碼的版本。您可以從 JTOpen 網站中下載程式碼並取得更多的資訊。

IBM Toolbox for Java JDBC 支援的加強功能

如需 JDBC 加強功能的相關資訊，請參閱第 291 頁的『V5R4 之 IBM Toolbox for Java JDBC 支援的加強功能』。如需新 JDBC 內容的相關資訊，請參閱第 295 頁的『IBM Toolbox for Java JDBC 內容』。

新類別

在 V5R3 之後的版本中，已新增下列類別。此處列出的所有類別都在 com.ibm.as400.access 套件中，除非另有指示。

- 第 27 頁的『BidiConversionProperties』
- 第 27 頁的『CallStackEntry』
- 第 52 頁的『IFSFileReader』
- 第 54 頁的『IFSFileWriter』
- 第 57 頁的『IFSSystemView』
- 第 58 頁的『ISeriesNetServer』
- 第 39 頁的『SaveFile』
- SignonHandler (介面)
- 第 40 頁的『Subsystem』

已強化的類別

下列類別的功能已得到顯著加強。此處列出的所有類別都在 com.ibm.as400.access 套件中，除非另有指示。

- 許多第 59 頁的『JDBC 類別』
- 第 49 頁的『IFSFile 類別』
- JarMaker 及 AS400ToolboxJarMaker (位於 utilities 資料包中)
- FTP 及 AS400FTP

已棄用的資料包

在 V5R3 之後的版本中，已棄用下列資料包。

- com.ibm.as400.vaccess
- com.ibm.as400.resource

已棄用的類別

在 V5R3 之後的版本中，已棄用下列類別。

- utilities.AS400ToolboxInstaller
- com.ibm.as400.access.NetServer 已由 第 58 頁的『ISeriesNetServer』 置換。
- com.ibm.as400.access.IFSTextFileInputStream 已由 第 52 頁的『IFSFileReader』 置換。
- com.ibm.as400.access.IFSTextFileOutputStream 已由 第 54 頁的『IFSFileWriter』 置換。

相容性

IBM Toolbox for Java 不再隨附 x4j400.jar (IBM XML 剖析器)。建議您在應用程式中使用下列其中一個與 JAXP 相容的 XML Parser：

- 內建於 JDK 1.4 及更新版本中的 XML Parser
- 可以從 xml.apache.org 取得的 Apache Xerces XML 剖析器
- /QIBM/ProdData/OS400/xml/lib 下之 i5/OS 隨附的 XML 剖析器之一

IBM Toolbox for Java 不再支援在 Netscape Navigator 或 Microsoft® Internet Explorer 的預設 JVM 中執行。若要執行能夠使用瀏覽器中 IBM Toolbox for Java 類別的 Applet，您必須安裝外掛程式，例如最新的 Sun Java 2 Runtime Environment (JRE) 外掛程式 。

IBM Toolbox for Java 不再包括 data400.jar。data400.jar 先前所含有的類別目前位於 jt400.jar 中。請從 CLASSPATH 陳述式中移除 data400.jar。

您無法使用此版次的 IBM Toolbox for Java，來解除序列化您使用 V5R1 之前之版次所序列化的部分物件。

如果使用 Secure Sockets Layer (SSL) 來將在用戶端與伺服器之間流通的資料加密，則您必須使用 Java Secure Socket Extension (JSSE)。

若要使用所有的 IBM Toolbox for Java 類別，請使用 Java 2 Platform 標準版 (J2SE)。使用 Vaccess 類別或 Graphical Toolbox 需要用到 J2SE 隨附的 Swing 套裝軟體。若要使用 PDML，您必須執行 Java Runtime Environment 1.4 版或更新版本。

如需詳細資訊，請複查執行 IBM Toolbox for Java 的 i5/OS 基本要求。

如何查看新增內容或變更內容

爲了協助您查看技術變更之處，此資訊使用：

- » 圖示，標示新增或變更資訊開始的位置。
- « 圖示，標示新增或變更資訊結束的位置。

如需本版次新增或變更功能的其他相關資訊，請參閱使用者備忘錄。

可列印的 PDF

檢視或下載 PDF 檔的 IBM Toolbox for Java 主題。您也可下載 IBM Toolbox for Java 種類的壓縮版套件。

若要檢視或下載 PDF 版本，請選取 IBM Toolbox for Java PDF (大約 6.7 MB)。

註: IBM Toolbox for Java 主題含有 PDF 檔中沒有的某些資訊。

儲存 PDF 檔

如欲在您的工作站上儲存 PDF，供檢視或列印之用：

- 在您的瀏覽器中以滑鼠右鍵按一下 PDF (以滑鼠右鍵按一下上述的鏈接)。
- 如果是使用 Internet Explorer，請按一下**另存目標...**。 如果是使用 Netscape Communicator，請按一下**另存鏈結...**。
- 瀏覽至要儲存 PDF 的目錄。
- 按一下**儲存**。

下載 Adobe Acrobat Reader

您需要 Adobe Acrobat Reader 來檢視或列印這些 PDF。您可以從 Adobe 網站 (www.adobe.com/products/acrobat/readstep.html) 下載複本。

以壓縮套件下載 IBM Toolbox for Java 資訊

您可以從 IBM Toolbox for Java 及 JTOpen 網站 ，下載內含 javadoc 的 IBM Toolbox for Java 主題壓縮套件。

註: 壓縮套件中的資訊鏈接至壓縮套件未包括的文件，所以這些鏈結無效。

安裝及管理 IBM Toolbox for Java

使用 IBM Toolbox for Java，可讓撰寫用戶端 Java Applet、Servlet 及存取 iSeries 資源、資料及程式的應用程式變得更容易。

管理 IBM Toolbox for Java 安裝

您只需要在要使用 IBM Toolbox for Java 的用戶端系統中安裝它，或者在用戶端可存取的網路位置上進行安裝。用戶端可以是個人電腦、專用工作站或 iSeries 系統。請不要忘記 iSeries 伺服器或伺服器的分割區可以配置成爲用戶端。在後者的情況下，您必須在伺服器的用戶端分割區中安裝 IBM Toolbox for Java。

您可以使用下列任一種方法 (單獨一種或組合) 來安裝與管理 IBM Toolbox for Java：

- 個別管理以安裝及個別管理每一個用戶端中的 IBM Toolbox for Java

- 網路管理單一安裝系統利用網路來安裝及管理伺服器中 IBM Toolbox for Java 的單一及共用安裝

下一節簡短說明每一種方式對於效能與管理的影響。您所選擇的 Java 應用程式開發以及資源管理方式，會決定您需要使用的方法 (或方法組合)。

個別的管理

您可以選擇分別管理個別用戶端中的 IBM Toolbox for Java 安裝。在個別用戶端中安裝 IBM Toolbox for Java 的主要優點，是可以縮減用戶端花在啓動使用 IBM Toolbox for Java 類別之應用程式的時間。

主要的缺點則是必須個別管理這些安裝。必須要有一個使用者或您建立的應用程式，來負責追蹤及管理安裝在每個工作站中的 IBM Toolbox for Java 版本。

單一安裝系統的網路管理

您也可以使用網路，在您所有用戶端都能存取的伺服器上安裝及管理單一 IBM Toolbox for Java 複本。這一種網路安裝系統具有下列優點：

- 所有用戶端都使用相同版本的 IBM Toolbox for Java
- 更新單一的 IBM Toolbox for Java 安裝，可以造福所有的用戶端
- 個別用戶端除了設定相同的起始 CLASSPATH 之外，沒有維護的問題

這種安裝也有缺點，它會增加用戶端花費在啓動 IBM Toolbox for Java 應用程式的時間。您也要將用戶端 CLASSPATH 指向該伺服器。您可以使用與 i5/OS 整合的 iSeries NetServer™，或使用讓您可以存取 iSeries 伺服器中檔案的不同方法，例如 iSeries Access for Windows。

安裝 IBM Toolbox for Java

IBM Toolbox for Java 的安裝方式取決於您要如何管理安裝系統。利用這些主題來安裝 IBM Toolbox for Java。

IBM Toolbox for Java 的 i5/OS 基本要求

請確定您的環境符合下列基本要求。

- 必要的 i5/OS 選項
- 與其它授權程式的相依關係
- 與不同層次 i5/OS 的相容性
- 在 i5/OS JVM 中執行時的原有最佳化
- 執行 ToolboxME for iSeries 應用程式的基本要求

註: 使用 IBM Toolbox for Java 之前，請先備妥與您環境相關的工作站基本要求。

必要的 i5/OS 選項:

若要在主從架構環境中執行 IBM Toolbox for Java，您必須啓用 QUSER 使用者設定檔、啓動主電腦伺服器及執行 TCP/IP。

- 必須啓用 QUSER 使用者設定檔，才能啓動主電腦伺服器。
- 主電腦伺服器接收並接受用戶端的連線要求。i5/OS 基本選項有包含「i5/OS 主電腦伺服器」選項 (授權產品 5722SS1)。如需詳細資訊，請參閱主電腦伺服器管理。
- 整合在 i5/OS 中的 TCP/IP 支援可讓您將伺服器連接至網路。如需詳細資訊，請參閱 TCP/IP。

啓動必要的 i5/OS 選項

請從 iSeries 指令行完成下列步驟，以啓動必要的 i5/OS 選項：

1. 確定已啓用 QUSER 設定檔。
2. 若要啓動 i5/OS 主電腦伺服器，請使用「啓動主電腦伺服器」CL 指令。鍵入 **STRHOSTSVR *ALL**，再按 **ENTER**。
3. 若要啓動 TCP/IP 分散式資料管理 (DDM) 伺服器，請使用「啓動 TCP/IP 伺服器」CL 指令。鍵入 **STRTCPSVR SERVER(*DDM)**，再按 **ENTER**。

檢查伺服器中是否已安裝 IBM Toolbox for Java:

許多 iSeries 伺服器都已經有安裝 IBM Toolbox for Java 授權產品。

若要查看是否已經安裝 IBM Toolbox for Java，請完成下列步驟：

1. 在「iSeries 領航員」中，選取並登入您要使用的系統。
2. 在**功能樹** (左窗格) 中，展開系統，然後展開**配置與服務**。
3. 展開**軟體**，然後展開**已安裝的產品**。
4. 在**明細**窗格 (右窗格) 中，查看 5722jc1 的**產品**直欄。如果您可以看到此產品，表示 IBM Toolbox for Java 授權程式已經安裝在所選取的伺服器中。

   註: 您也可以透過使用「跳至功能表」CL 指令 (**GO MENU(LICPGM)**)，然後使用選項 11，來檢查是否已安裝 IBM Toolbox for Java。

如果尚未安裝 IBM Toolbox for Java，您可以安裝 IBM Toolbox for Java 授權產品。

如果已安裝了舊版的 IBM Toolbox for Java，請先刪除目前安裝的版本，然後再安裝 IBM Toolbox for Java 授權產品。爲了避免可能發生的問題，請在刪除目前安裝的版本之前，先考慮備份目前安裝的 IBM Toolbox for Java 版本。

檢查 QUSER 設定檔:

「i5/OS 主電腦伺服器」是在 QUSER 使用者設定檔下啓動的，所以您首先必須確定 QUSER 設定檔已經啓用。

檢查 QUSER 設定檔

要使用指令行來檢查 QUSER 設定檔，請完成下列步驟：

1. 在 iSeries 指令行中，鍵入 `DSPUSRPRF USRPRF(QUSER)`，然後按 **Enter** 鍵。
2. 確定 **Status** 是 *ENABLED。如果設定檔的狀態不是 *ENABLED，請變更 QUSER 設定檔。

變更 QUSER 使用者設定檔:

如果 QUSER 設定檔不是 *ENABLED，則必須加以啓用才能啓動「i5/OS 主電腦伺服器」。此外，QUSER 設定檔密碼也不可爲 *NONE。如果此爲 TRUE，則需重設它。

若要使用指令行來啓用 QUSER 設定檔，請完成下列步驟：

1. 鍵入 `CHGUSRPRF USRPRF(QUSER)`，然後按 **ENTER** 鍵。
2. 將**狀態**欄位變更爲 *ENABLED，並按 **ENTER** 鍵。

QUSER 使用者設定檔現已備妥，可啓動「i5/OS 主電腦伺服器」。

與其它授權程式的相依關係:

根據您使用 IBM Toolbox for Java 的方式，您可能必須安裝其他授權程式。

排存檔檢視器

若要使用 IBM Toolbox for Java 的排存檔檢視器功能 (SpooledFileViewer 類別)，請確定您已經在伺服器中安裝主電腦選項 8 (AFP™ 相容字型)。

註： 只有在連接到 V4R4 或更新的系統時， SpooledFileViewer、PrintObjectPageInputStream 及 PrintObjectTransformedInputStream 類別才能作用。

Secure Sockets Layer

若要使用 Secure Sockets Layer (SSL)，請確定您已經安裝好下列項目：

- IBM HTTP Server for iSeries 授權程式 5722-DG1
- i5/OS 選項 34 (數位憑證管理程式)
- IBM Cryptographic Access Provider for iSeries 128 位元 5722-AC3 (僅適用於 V5R4 之前的版次)

如需相關 SSL 詳細資訊，請參閱 第 315 頁的『Secure Sockets Layer 及 Java Secure Socket Extension』。

用來使用 Applet、Servlet 或 SSL 的 HTTP 伺服器

如果您要在 iSeries 伺服器上使用 Applet、Servlet 或 SSL，必須設定一個 HTTP 伺服器，並在 iSeries 伺服器中安裝類別檔。如需 IBM HTTP Server 的相關資訊，請參閱 IBM HTTP Server for AS/400® Webmaster's Guide (GC41-5434)，網址如下： http://www.ibm.com/eserver/iseries/products/http/docs/doc.htm  。 Webmaster's Guide 有提供 HTML 與 PDF 兩種格式。

如需「數位憑證管理程式」及如何透過 IBM HTTP Server 建立及使用數位憑證的相關資訊，請參閱數位憑證管理。

與不同層次 i5/OS 的相容性:

因爲伺服器及用戶端上都可以使用 IBM Toolbox for Java，所以相容性的問題就會影響到伺服器中的運作情形以及由用戶端連接回伺服器的情況。

在伺服器中執行 IBM Toolbox for Java

若要在 iSeries 系統中安裝 IBM Toolbox for Java (授權程式 5722-JC1 V5R4M0)，該伺服器中必須執行下列其中一個版本：

- i5/OS 版本 5 版次 4
- i5/OS 版本 5 版次 3
- i5/OS 版本 5 版次 2

系統上只能安裝一種 IBM Toolbox for Java 授權程式版本。若要安裝不同的版本，請先移除現有的 IBM Toolbox for Java 授權程式。

使用 IBM Toolbox for Java，由用戶端連接回伺服器

您可以在用戶端與所連接的伺服器中使用不同版本的 IBM Toolbox for Java。若要使用 IBM Toolbox for Java 來存取 iSeries 系統中的資料及資源， **您所連接的伺服器**必須執行下列其中一個版本：

- i5/OS 版本 5 版次 4
- i5/OS 版本 5 版次 3
- i5/OS 版本 5 版次 2

下表說明在 i5/OS 中安裝 IBM Toolbox for Java，及連接回不同的版本時，所需考慮的相容性基本要求。

註: IBM Toolbox for Java 不支援向前相容性。您無法將 IBM Toolbox for Java 安裝在 (或用它來連接到) 一個執行更新版 i5/OS 的伺服器上。例如，若您所使用的 IBM Toolbox for Java 隨附於 i5/OS V5R2，則不能將它安裝在 (或連接到) 執行 i5/OS V5R4 的伺服器上。

LPP	i5/OS 隨附	安裝在 i5/OS 中	連接回 i5/OS
5722-JC1 V5R2M0	V5R2	V4R5 與以上的版本	V4R5 與以上的版本
5722-JC1 V5R3M0	V5R3	V5R1 與以上的版本	V5R1 與以上的版本
5722-JC1 V5R4M0	V5R4	V5R2 與以上的版本	V5R2 與以上的版本

在 iSeries JVM 中執行時的原有最佳化:

原有最佳化是一組功能，能讓 IBM Toolbox for Java 類別在 i5/OS 中的運作情形符合使用者的預期。最佳化只有在 iSeries JVM 中執行時，才會影響 IBM Toolbox for Java 的作業。

請記得，當您使用的 IBM Toolbox for Java 版本與伺服器中的 i5/OS 版本相符時，您的 Java 程式才會使用原有最佳化。最佳化包括：

- 登入：AS400 物件中沒有指定使用者 ID 或密碼時，就會使用目前工作的使用者 ID 與密碼
- 直接呼叫 i5/OS API，而不對主電腦伺服器進行 Socket 呼叫：
  - 符合安全基本要求時，就呼叫記錄層次資料庫存取、資料佇列及使用者空間。
  - 符合安全基本要求與緒安全基本要求時，則進行程式呼叫以及指令呼叫。

註: 當 Java 程式與資料庫檔案都位於相同的 iSeries 伺服器上時，爲了取得最佳效能，請將 JDBC 驅動程式內容設爲使用原有的驅動程式。

您不需要爲了取得最佳化而變更 Java 應用程式。IBM Toolbox for Java 會在適當時自動啓用最佳化。

原有最佳化相容性基本要求

下表說明您必須執行哪一種版本的 IBM Toolbox for Java LPP 與 i5/OS，才能使用原有的最佳化。此表格僅引證會影響到原有最佳化的相容性問題。有關一般的相容性問題，請參閱與不同層次 i5/OS 的相容性。

i5/OS 的層次	使用原有最佳化所需的 IBM Toolbox for Java LPP
V5R2	5722-JC1 V5R2M0
V5R3	5722-JC1 V5R3M0
V5R4	5722-JC1 V5R4M0

爲了提高效能，請務必使用包含 i5/OS 原有最佳化的 Jar 檔。如需詳細資訊，請參閱 jar 檔中的附註 1。

若 IBM Toolbox for Java 與 i5/OS 的版本不相符，將無法使用原有最佳化。在此情況下，IBM Toolbox for Java 的運作狀況將如同是在用戶端中執行一般。

ToolboxME for iSeries 基本要求:

您的工作站、無線裝置及伺服器必須符合以下列出的特定基本要求，才能開發與執行 ToolboxME for iSeries 應用程式。雖然 IBM Toolbox for Java 2 Micro Edition 被視爲 IBM Toolbox for Java 的一部分，但授權產品中並未將其包含在內。

ToolboxME for iSeries (jt400Micro.jar) 包含於 Toolbox for Java 的開放原始碼版本中，稱爲 JTOpen。您必須個別下載及設定 ToolboxME for iSeries (包含在 JTOpen 中)。

基本要求

若要使用 ToolboxME for iSeries，您的工作站、Tier0 無線裝置及伺服器必須符合下列基本要求。

工作站基本要求

開發 ToolboxME for iSeries 應用程式的工作站基本要求：

- Java 2 Platform 標準版 1.3 版或更高版本
- 無線裝置的 Java 虛擬機器
- 無線裝置模擬程式

無線裝置基本要求

在 Tier0 裝置中執行 ToolboxME for iSeries 應用程式的唯一基本要求爲：使用無線裝置的 Java 虛擬機器。

伺服器基本要求

使用 ToolboxME for iSeries 應用程式的伺服器基本要求：

- IBM Toolbox for Java 或最新版 JTOpen 所隨附的 MEServer 類別
- IBM Toolbox for Java 的 i5/OS 基本要求

IBM Toolbox for Java 的工作站基本要求

請確定您的工作站符合下列基本要求。

註: 使用 IBM Toolbox for Java 之前，請先備妥與您環境相關的 i5/OS 基本要求。

執行 IBM Toolbox for Java 應用程式的工作站基本要求:

若要開發與執行 IBM Toolbox for Java 應用程式，請確定您的工作站符合下列基本要求。

- 我們建議您使用受支援的 Java 2 標準版 (J2SE) Java 虛擬機器。許多新的 IBM Toolbox for Java 功能需要使用 JVM 1.4 版或更高版本。
- 使用 Vaccess 類別或 Graphical Toolbox 需要用到 J2SE 隨附的 Swing。您也可以從 Sun 的 Java Foundation Classes  網站上下載 Swing 1.1。已經測試的環境如下：
  - Windows 2000
  - Windows XP
  - AIX® 4.3.3.1 版
  - Sun Solaris 5.7 版
  - i5/OS 版本 5 版次 2 或更新版本
  - Linux® (Red Hat 7.0)
- 已安裝及配置 TCP/IP

執行 IBM Toolbox for Java Applet 的工作站基本要求:

若要開發與執行 IBM Toolbox for Java 應用程式，請確定您的工作站符合下列基本要求。

- 具有相容之 Java 虛擬機器 (JVM) 的瀏覽器。已經測試的環境如下：

– Netscape Communicator 4.7，使用 Java 1.3 或更新版本的外掛程式

註: IBM Toolbox for Java 不再支援在 Netscape Navigator 或 Microsoft Internet Explorer 的預設 JVM 中執行。對於使用 IBM Toolbox for Java 類別要在瀏覽器上執行的 Applet，您應該爲它們安裝外掛程式，如 Sun Java 2 Runtime Environment (JRE) plug-in  。

- 已安裝及配置 TCP/IP
- 工作站必須連接至執行 i5/OS V5R2 或更新版本的伺服器

IBM Toolbox for Java 的工作站 Swing 基本要求:

IBM Toolbox for Java 從 V4R5 變更爲支援 Swing 1.1，此一版次會繼續支援。切換至 Swing 需要變更 IBM Toolbox for Java 類別中的程式設計。所以，如果您的程式使用 Graphical Toolbox 或 V4R5 版以前的 Vaccess 類別，您的程式也需要變更。

除了程式設計變更之外，當執行程式時，Swing 類別必須位於 CLASSPATH 中。Swing 類別是 Java 2 Platform 的一部分。如果您沒有 Java 2 Platform，可以從 Sun Microsystems 公司下載 Swing 1.1 類別。 

在 iSeries 伺服器中安裝 IBM Toolbox for Java

僅當您將伺服器或伺服器分割區配置成用戶端時，才需要在 iSeries 伺服器中安裝 IBM Toolbox for Java。

註: 原有版本的 IBM Toolbox for Java 隨附於 i5/OS。所以，如果您只是要在 iSeries 伺服器中使用 IBM Toolbox for Java，就不需要安裝授權產品。如需原有版本 IBM Toolbox for Java 的相關資訊，請參閱 Jar 檔：附註 1。

安裝 IBM Toolbox for Java 之前，您必須確定您的 i5/OS 版本符合執行 IBM Toolbox for Java 的基本要求。此外，有些伺服器會預先配置爲已安裝 IBM Toolbox for Java。您可以判定伺服器中是否已安裝了 IBM Toolbox for Java 授權產品。

安裝 IBM Toolbox for Java

您可以使用「iSeries 領航員」或指令行來安裝 IBM Toolbox for Java 授權程式。

使用 iSeries 領航員安裝 IBM Toolbox for Java

若要使用「iSeries 領航員」安裝 IBM Toolbox for Java，請完成下列步驟：

1. 在「iSeries 領航員」中，登入您要使用的系統。
2. 在「功能樹」(左窗格) 中，展開**我的連接**。
3. 在**我的連線**下，以滑鼠右鍵按一下您要安裝 IBM Toolbox for Java 的系統。
4. 選取**執行指令**。
5. 在**復置授權程式 (RSTLICPGM)**對話框中，鍵入下列資訊，然後按一下**確定**：
   - 產品：5722JC1
   - 裝置：裝置或儲存檔案名稱

   註: 如需詳細資訊，請按一下**復置授權程式 (RSTLICPGM)**對話框中的**說明**。

完成下列步驟，就可以使用「iSeries 領航員」來檢視所產生之「管理中心指令」作業的狀態：

1. 展開**管理中心**。
2. 展開**作業活動**。

3. 在**作業活動**底下，選取**指令**。
4. 在「明細」窗格中，按一下適當的**執行指令**作業。

使用指令行來安裝 IBM Toolbox for Java

若要從 iSeries 指令行安裝 IBM Toolbox for Java，請完成下列步驟：

1. 在 iSeries 指令行上，使用「跳至功能表」CL 指令。鍵入 **GO MENU(LICPGM)** 再按 **ENTER**。
2. 選取 **11.安裝授權程式**。
3. 選取 **5722-JC1 IBM Toolbox for Java**。

有關安裝授權程式的詳細資訊，請參閱管理軟體及授權程式。

在工作站中安裝 IBM Toolbox for Java

安裝 IBM Toolbox for Java 之前，請先備妥與您環境相關的工作站基本要求。

在工作站中安裝 IBM Toolbox for Java 的方式取決於您要如何管理安裝系統：

- 若要在個別的用戶端中安裝 IBM Toolbox for Java，請將 JAR 檔複製到您的工作站中，並配置工作站 CLASSPATH。
- 若要使用安裝於伺服器中的 IBM Toolbox for Java，您只需要配置工作站 CLASSPATH 來指向伺服器安裝即可。若要將工作站 CLASSPATH 指向伺服器，您的伺服器必須有安裝 iSeries Netserver。

本文件說明如何將類別檔案複製到工作站中。如需在工作站中設定 CLASSPATH 的相關資訊，請參閱工作站的作業系統文件，或位於 Sun Java 網站 上的資訊。

註: 您的系統還必須符合 i5/OS 的基本要求，才能在應用程式中使用 IBM Toolbox for Java 類別。

IBM Toolbox for Java 類別檔案封裝在數個 Jar 檔中，因此您必須複製一或多個這類 JAR 檔到您的工作站中。如需特定 IBM Toolbox for Java 功能需要哪些 Jar 檔的相關資訊，請參閱 Jar 檔。

範例：複製 jt400.jar

下列範例假設您要複製 jt400.jar，此檔包含核心 IBM Toolbox for Java 類別。

若要手動複製 JAR 檔，請完成下列步驟：

1. 在下面的目錄中找出 jt400.jar 檔：/QIBM/ProdData/HTTP/Public/jt400/lib
2. 將 jt400.jar 從伺服器複製到您的工作站。有不同的方法可用來完成此項步驟：
   - 使用 iSeries Access for Windows，將工作站中的網路磁碟機對映至伺服器，然後複製檔案。
   - 使用檔案轉送通訊協定 (FTP) 將檔案傳送 (以二進位模式) 到工作站。
3. 更新工作站的 CLASSPATH 環境變數。
   - 例如，如果您在使用 Windows NT®，且您已將 jt400.jar 複製到 C:\jt400\lib，請在 CLASSPATH 的末端新增下列字串：

     ```
     ;C:\jt400\lib\jt400.jar
     ```

您也可以選擇使用開放原始碼版本的 IBM Toolbox for Java (稱爲 JTOpen)。如需 JTOpen 的相關資訊，請參閱 IBM Toolbox for Java 與 JTOpen 網站 。

Jar 檔:

IBM Toolbox for Java 是以一組 jar 檔的形式提供的。每個 jar 檔包含提供特定功能的 Java 套件。您可以減少所需儲存體空間的容量，其方法爲只使用需要的 jar 檔來啓用您要的特定功能。

若要使用 jar 檔，請確定您的 CLASSPATH 已包含了它的登錄。

下圖指出您必須將哪些 JAR 檔新增到 CLASSPATH， 才能使用相關的功能或套件。

IBM Toolbox for Java 套件或功能	**Jar 檔必須在您的 CLASSPATH 中**
Access 類別	jt400.jar (用戶端) 或 jt400Native.jar (伺服器) 附註 1，或 proxy 環境中的 jt400Proxy.jar
第 226 頁的『CommandHelpRetriever 類別』	jt400.jar (用戶端) 或 jt400Native.jar (伺服器) 附註 1 以及 XML 剖析器 和 XSLT 處理器附註 2
CommandPrompter 附註 3	jt400.jar、jui400.jar、util400.jar 附註 4 和 XML 剖析器附註 2
Commtrace 類別	jt400.jar (用戶端) 或 jt400Native.jar (伺服器) 附註 1
HTML 類別	jt400.jar 附註 1 加上 jt400Servlet.jar (用戶端)，或 jt400Native.jar (伺服器) 附註 1
HTMLDocument 類別	HTML 類別所需要的相同 jar，加上 XML 剖析器和 XSLT 處理器附註 2
JCA 類別	jt400.jar (用戶端) 或 jt400Native.jar (伺服器) 附註 1
JDBC 資料來源 GUI	jt400.jar (用戶端) 附註 1 和 jui400.jar附註 5
NLS 系統和錯誤訊息	jt400Mri_lang_cntry.jar 附註 6
PCML (開發和執行時間，已剖析) 附註 7	jt400.jar (用戶端) 或 jt400Native.jar (伺服器) 附註 1、 附註 8 和 XML 剖析器附註 2
PCML (執行期間，已編序)	jt400.jar (用戶端) 或 jt400Native.jar (伺服器) 附註 1、 附註 8
PDML (開發) 附註 3	uitools.jar、jui400.jar、util400.jar 附註 4 和 XML 剖析器附註 2
PDML (執行時間，已剖析) 附註 3	jui400.jar、util400.jar 附註 4 和 XML 剖析器附註 2
PDML (執行時間，已序列化) 附註 3	jui400.jar 和 util400.jar 附註 4
ReportWriter 類別	jt400.jar (用戶端) 或 jt400Native.jar (伺服器) 附註 1、 reportwriter jar 附註 9，以及 XML 剖析器和 XSLT 處理器 附註 2
資源類別	jt400.jar (用戶端) 或 jt400Native.jar (伺服器) 附註 1
RFML	jt400.jar (用戶端) 或 jt400Native.jar (伺服器) 附註 1，以及 XML 剖析器附註 2
Security 類別	jt400.jar (用戶端) 或 jt400Native.jar (伺服器) 附註 1，或 proxy 環境中的 jt400Proxy.jar
Servlet 類別	jt400.jar 附註 1 加上 jt400Servlet.jar (用戶端)，或 jt400Native.jar (伺服器) 附註 1
iSeries 系統除錯器 附註 3	jt400.jar (用戶端) 附註 1 和 tes.jar
ToolboxME for iSeries	jt400Micro.jar (用戶端) 附註 10 和 jt400.jar (伺服器) 或 jt400Native.jar (伺服器)附註 1
Vaccess 類別	jt400.jar (用戶端) 附註 1
XPCML	jt400.jar (用戶端) 或 jt400Native.jar (伺服器) 附註 1，以及 XML 剖析器 和 XSLT 處理器附註 2

附註 1：有些 IBM Toolbox for Java 類別位於一個以上的 jar 檔之中：

- **jt400.jar** - Access、commtrace、JCA、JDBC 支援、MEServer、PCML、資源、RFML、安全、公用程式、vaccess 和 XPCML。
- **jt400.zip** - 使用 jt400.jar 代替 jt400.zip。jt400.zip 可用來保持與前版次 IBM Toolbox for Java 的相容性。
- **jt400Access.zip** - jt400.jar 中 vaccess 以外的所有類別。jtAccess400.zip 可用來保持與前版次 IBM Toolbox for Java 的相容性。請使用 jt400.jar 或 jt400Native.jar 代替 jt400Access.zip。
- **jt400Native.jar** - Access、HTML、MEServer、PCML、資源、 RFML、安全、XPCML 和原有的最佳化。原有的最佳化是指一組在 iSeries JVM 上執行時會利用 iSeries 功能的類別 (小於 20)。因爲 jt400Native.jar 包含原有的最佳化，所以在 iSeries JVM 上執行時，請使用 jt400Native.jar 代替 jt400.jar。jt400Native.jar 隨附於 i5/OS，且位於目錄 /QIBM/ProdData/OS400/jt400/lib 中。
- **jt400Native11x.jar** - 使用 jt400Native.jar 代替 jt400Native11x.jar。 jt400Native11x.jar 可用來保持與前版次 IBM Toolbox for Java 的相容性。

附註 2：必須使用 XML 剖析器或 XSLT 處理器時，請確定它們可與 JAXP 相容。詳細資訊，請參閱下一頁：

第 370 頁的『XML 剖析器與 XSLT 處理器』

附註 3：若要使用 CommandPrompter、PDML，或「iSeries 系統除錯程式」，也必須要有 IBM Toolbox for Java 以外的另一個 jar 檔：jhall.jar。關於下載 jhall.jar 的詳細資訊，請參閱 Sun JavaHelp(TM) 網站 。

附註 4：util400.jar 包含一些 iSeries 專有類別，可供格式化輸入及使用指令行提示器之用。使用 CommandPrompter 類別需要 util400.jar。使用 PDML 時不一定要有 util400.jar， 但它是有用的。

附註 5：jui400.jar 包含使用 JDBC DataSource GUI 介面時需要的類別。jt400.jar (附註 1) 包含所有其他 JDBC 功能需要的類別。

附註 6：jt400Mri_xx_yy.jar 包含一些轉譯訊息，包括異常訊息、對話框以及另一個正常處理程序的輸出所含的字串。在 jt400Mri_lang_cntry.jar 中， lang = 「ISO 語言碼」，cntry = 「ISO 國家或地區碼」，用來轉譯內含文字。在某些情況下，不會使用「ISO 國家或地區碼」。在 iSeries 上安裝 IBM Toolbox for Java 授權程式的特定國家語言版本，就會同時安裝適當的 jt400Mri_lang_cntry.jar 檔。如果該語言不受支援，則會安裝預設的英文版本，其包含於 IBM Toolbox for Java JAR 檔中。

- 例如，安裝德國語言版本的授權程式 5722-JC1 會安裝德文 jar 檔 jt400Mri_de.jar。

將這些 JAR 檔多加入幾個到 classpath 中，即可增加其他語言的支援。Java 會根據目前的語言環境載入正確的字串。

附註 7：在開發期間序列化 PCML 檔有兩個好處：

1. 只有在開發期間才需要剖析 PCML 檔，執行期間不需要
2. 使用者在他們 CLASSPATH 中只要有較少的 jar 檔即可執行應用程式

若要在開發期間剖析 PCML 檔，則在 data.jar 或 jt400.jar 中必須有 PCML 執行時間，在 x4j400.jar 中必須有 PCML 剖析器。若要執行編序的應用程式， 使用者只需要 jt400.jar。如需相關資訊，請參閱使用 PCML 建置 iSeries 程式呼叫。

附註 8：使用 jt400.jar 和 jt400Native.jar，而非 data400.jar。data400.jar 包含一些 PCML 執行時間類別，這些類別目前也位於 jt400.jar 和 jt400Native.jar 中 (附註 1)。data400.jar 可用來保持與前版次 IBM Toolbox for Java 的相容性。

附註 9：不只一個 JAR 檔有 ReportWriter 類別副本：

- composer.jar

- outputwriter.jar
- reportwriters.jar

如果應用程式將 PCL 資料直接傳送到 iSeries 排存檔，您必須使用適當的 JAR 檔，讓存取類別成爲可用的類別 (附註 1)。建立一個排存檔來保留 PCL 資料需要 AS400、OutputQueue、PrintParameterList 及 SpooledFileOutputStream 類別。 詳細資訊，請參閱 ReportWriter 類別。

附註 10：jt400Micro.jar 不含執行 MEServer 時需要的類別， 這些類別位於 jt400.jar 和 jt400Native.jar 中 (附註 1)。只有 IBM Toolbox for Java and JTOpen 網站  才提供 jt400Micro.jar。

系統內容

您可指定系統內容，以配置 IBM Toolbox for Java 的各個層面。

例如，您可使用系統內容來定義 PROXY 伺服器或追蹤層次。系統內容對於簡便的執行時間配置非常有用，而不需重新編譯程式碼。當您在執行時間變更系統內容，系統內容的作業就像環境變數，變更通常要等到下次執行應用程式時才會反映。

您可使用下列數種方式設定系統內容：

- **使用 java.lang.System.setProperties() 方法**

  您可使用 java.lang.System.setProperties() 方法，程式化地設定系統內容。

  例如，下列程式碼將 com.ibm.as400.access.AS400.proxyServer 內容設定爲 hqoffice：

  ```
  Properties systemProperties = System.getProperties();
     systemProperties.put ("com.ibm.as400.access.AS400.proxyServer", "hqoffice");
      System.setProperties (systemProperties);
  ```

- **使用 java 指令的 -D 選項**

  許多環境可讓您使用 Java 指令的 -D 選項， 在從指令行執行應用程式時設定系統內容。

  例如，下列程式執行名爲 Inventory 的應用程式， 且其 com.ibm.as400.access.AS400.proxyServer 內容設定 爲 hqoffice：

  ```
  java -Dcom.ibm.as400.access.AS400.proxyServer=hqoffice Inventory
  ```

- **使用 jt400.properties 檔案**

  在部分環境中，可能不適宜指示所有的使用者設定其自有的系統內容。另一個選擇方案爲在名稱爲 jt400.properties 的檔案中指定 IBM Toolbox for Java 系統內容，將該檔案視爲 com.ibm.as400.access 套件的一部分加以搜尋。換言之， 將 jt400.properties 檔案置於 classpath 所指向的 com/ibm/as400/access 目錄。

  例如，在 jt400.properties 檔案中插入下列程式行，以將 com.ibm.as400.access.AS400.proxyServer 內容設定爲 hqoffice：

  ```
  com.ibm.as400.access.AS400.proxyServer=hqoffice
  ```

  反斜線字元 (\) 的功能如同內容檔案中的跳出字元。 使用雙反斜線 (\\) 指定文字反斜線字元。

  針對您的環境修改這個 jt400.properties 檔案的範例。

- **使用 Properties 類別**

部分瀏覽器不載入沒有明確變更安全性設定值的內容檔案。不過，大部分的瀏覽器都接受 .class 檔案中的內容，因此，也可使用延伸 java.util.Properties 的 com.ibm.as400.access.Properties 類別來指定 IBM Toolbox for Java 系統內容。

例如，若要將 com.ibm.as400.access.AS400.proxyServer 內容設爲 hqoffice，請使用下列 Java 程式碼：

```
package com.ibm.as400.access;

public class Properties
extends java.util.Properties
{
    public Properties ()
   {
      put ("com.ibm.as400.access.AS400.proxyServer", "hqoffice");
   }
}
```

針對您的環境，修改及編譯此 Properties.java 原始檔的範例。

如果使用上述說明的多個機制來設定 IBM Toolbox for Java 系統內容，則其優先順序如下 (依遞減的優先順序)：

1. 使用 java.lang.System.setProperties() 程式化地設定系統內容
2. 使用 Java 指令的 -D 選項設定系統內容
3. 使用 Properties 類別設定系統內容
4. 使用 jt400.properties 檔案設定系統內容

IBM Toolbox for Java 支援下列的系統內容：

- 『Proxy 伺服器內容』
- 第 15 頁的『追蹤內容』
- 第 15 頁的『CommandCall/ProgramCall 內容』
- 第 15 頁的『FTP 內容』
- 第 16 頁的『連線內容』

Proxy 伺服器內容

PROXY 伺服器內容	說明
com.ibm.as400.access.AS400.proxyServer	使用下列格式指定 PROXY 伺服器主電腦名稱及埠號： hostName:portNumber 埠號是可選用的。
com.ibm.as400.access.SecureAS400.proxyEncryptionMode	使用 SSL 指定要加密的虛擬資料流程部分。有效值爲： • 1 = Proxy 用戶端到 PROXY 伺服器 • 2 = Proxy 伺服器到 iSeries • 3 = Proxy 用戶端到 proxy 伺服器及 proxy 伺服器到 iSeries
com.ibm.as400.access.TunnelProxyServer.clientCleanupInterval	指定 PROXY 伺服器尋找閒置中連線的頻率 (以秒爲單位)。PROXY 伺服器啓動一個緒來尋找不再通訊的用戶端。使用此內容來設定緒尋找閒置中連線的頻率。

PROXY 伺服器內容	說明
com.ibm.as400.access.TunnelProxyServer.clientLifetime	指定在 PROXY 伺服器移除對物件的參照之前，用戶端可閒置的時間 (以秒爲單位)， 使 JVM 可進行垃圾收集。PROXY 伺服器啓動一個緒來尋找不再通訊的用戶端。使用此內容以設定在用戶端上執行垃圾收集之前的閒置時間。

追蹤內容

追蹤內容	說明
com.ibm.as400.access.Trace.category	指定要啓用的追蹤種類。這是一個包含任何追蹤種類組合的逗點分界式清單。完整的追蹤種類清單定義於 Trace 類別中。
com.ibm.as400.access.Trace.file	指定追蹤輸出寫入的檔案。預設爲將追蹤輸出寫入 System.out。
com.ibm.as400.access.ServerTrace.JDBC	指定要在 JDBC 伺服器工作上啓動的追蹤種類。有關支援值的詳細資訊，請參閱 JDBC 伺服器追蹤內容。

CommandCall/ProgramCall 內容

CommandCall/ProgramCall 內容	說明
com.ibm.as400.access.CommandCall.threadSafe	指定是否會將 CommandCall 假設爲安全緒。若爲 `true`， 則假設所有的 CommandCall 爲安全緒。若爲 `false`， 則假設所有的 CommandCall 爲非安全緒。如果已於所指定的 CommandCall 物件上執行 `CommandCall.setThreadSafe(true/false)` 或 `AS400.setMustUseSockets(true)`，則系統不處理此內容。
com.ibm.as400.access.ProgramCall.threadSafe	指定是否會將 ProgramCall 假設爲安全緒。若爲 `true`， 則假設所有的 ProgramCall 爲安全緒。若爲 `false`， 則假設所有的 ProgramCall 爲非安全緒。如果已於所指定的 ProgramCall 物件上執行 `CommandCall.setThreadSafe(true/false)` 或 `AS400.setMustUseSockets(true)`，則系統不處理此內容。

FTP 內容

FTP 內容	說明
com.ibm.as400.access.FTP.reuseSocket	處於「作用中」模式時，指定是否重覆使用 Socket，以轉送多個檔案 (透過單一 FTP 實例)。如果爲 `true`，則會重覆使用 Socket。如果爲 `false`，則會對每個檔案轉送建立一個新的 Socket。如果已對所指定的 FTP 物件執行 `FTP.setReuseSocket(true/false)`，則系統不處理該物件的此內容。

連線內容

連線內容	說明
com.ibm.as400.access.AS400.signonHandler	指定預設登入處理程式。如果已對所指定的 AS400 物件執行 AS400.setSignonHandler() 或已呼叫 AS400.setDefaultSignonHandler()，則系統不會處理該物件的此內容。

範例：內容檔

```
#=========================================================#
# IBM Toolbox for Java                                    #
#---------------------------------------------------------#
# Sample properties file                                  #
#                                                         #
# Name this file jt400.properties and store it in a       #
# com/ibm/as400/access directory that is pointed to by    #
# the classpath.                                          #
#=========================================================#

#---------------------------------------------------------#
# Proxy server system properties                          #
#---------------------------------------------------------#

# This system property specifies the proxy server host name
# and port number, specified in the format: hostName:portNumber
# The port number is optional.
com.ibm.as400.access.AS400.proxyServer=hqoffice

# This system property specifies which portion of the proxy
# data flow is encrypted via SSL.  Valid values are:
#   1 - Proxy client to proxy server
#   2 - Proxy server to AS/400
#   3 - Proxy client to proxy, and proxy server to AS/400
com.ibm.as400.access.SecureAS400.proxyEncryptionMode=1

# This system property specifies how often, in seconds,
# the proxy server will look for idle connections. The
# proxy server starts a thread to look for clients that are
# no longer communicating. Use this property to set how
# often the thread looks for idle connections.
com.ibm.as400.access.TunnelProxyServer.clientCleanupInterval=7200

# This system property specifies how long, in seconds, a
# client can be idle before it is cleaned up. The proxy server
# starts a thread to look for clients that are no longer
# communicating. Use this property to set long a client can
# be idle before it is cleaned up.
com.ibm.as400.access.TunnelProxyServer.clientLifetime=2700


#---------------------------------------------------------#
# Trace system properties                                 #
#---------------------------------------------------------#

# This system property specifies which trace categories to enable.
# This is a comma-delimited list containing any combination of trace
# categories.  The complete list of trace categories is defined in
# the Trace class.
com.ibm.as400.access.Trace.category=error,warning,information

# This system property specifies the file to which trace output
```

```
# is written.  The default is to write trace output to System.out.
com.ibm.as400.access.Trace.file=c:\\temp\\trace.out


#---------------------------------------------------------#
# Command Call system properties                          #
#---------------------------------------------------------#

# This system property specifies whether CommandCalls should
# be assumed to be thread-safe. If true, all CommandCalls are
# assumed to be thread-safe. If false, all CommandCalls are
# assumed to be non-thread-safe. This property is ignored
# for a given CommandCall object if either
# CommandCall.setThreadSafe(true/false) or
# AS400.setMustUseSockets(true) has been performed on the object.
com.ibm.as400.access.CommandCall.threadSafe=true


#---------------------------------------------------------#
# Program Call system properties                          #
#---------------------------------------------------------#

# This system property specifies whether ProgramCalls should
# be assumed to be thread-safe. If true, all ProgramCalls are
# assumed to be thread-safe. If false, all ProgramCalls are
# assumed to be non-thread-safe. This property is ignored
# for a given ProgramCall object if either
# ProgramCall.setThreadSafe(true/false) or
# AS400.setMustUseSockets(true) has been performed on the object.
com.ibm.as400.access.ProgramCall.threadSafe=true


#---------------------------------------------------------#
# FTP system properties                                   #
#---------------------------------------------------------#

# This system property specifies whether the socket is reused
# for multiple file transfers (through a single FTP instance),
# when in "active" mode.
# If true, the socket is reused. If false, a new socket is
# created for each file transfer.
# This property is ignored for a given FTP object if
# FTP.setReuseSocket(true/false) has been performed on the object.
com.ibm.as400.access.FTP.reuseSocket=true


#---------------------------------------------------------#
# Connection system properties                            #
#---------------------------------------------------------#

# This system property specifies the default signon handler.
# This property is ignored for a given AS400 object if
# AS400.setSignonHandler() has been performed on
# the object, or if AS400.setDefaultSignonHandler()
# has been called.
com.ibm.as400.access.AS400.signonHandler=mypackage.MyHandler


# End
```

範例：系統內容類別原始檔

```
//============================================================
// IBM Toolbox for Java
//------------------------------------------------------------
// Sample properties class source file
//
```

```
// Compile this source file and store the class file in
// the classpath.

//=========================================================
package com.ibm.as400.access;

public class Properties
extends java.util.Properties
{
   public Properties ()
   {
      /*---------------------------------------------------------*/
      /* Proxy server system properties*/
      /*---------------------------------------------------------*/

      // This system property specifies the proxy server host name
      // and port number, specified in the format: hostName:portNumber
      // The port number is optional.
      put ("com.ibm.as400.access.AS400.proxyServer", "hqoffice");

      // This system property specifies which portion of the proxy
      // data flow is encrypted via SSL.  Valid values are:
      // 1 - Proxy client to proxy server
      //   2 - Proxy server to iSeries server
      //   3 - Proxy client to proxy, and proxy server to iSeries server
      put("com.ibm.as400.access.SecureAS400.proxyEncryptionMode", "1");

      // This system property specifies how often, in seconds,
      // the proxy server will look for idle connections. The
// proxy server starts a thread to look for clients that are
      // no longer communicating. Use this property to set how
// often the thread looks for idle connections.

      put("com.ibm.as400.access.TunnelProxyServer.clientCleanupInterval", "7200");

      // This system property specifies how long, in seconds, a
      // client can be idle before it is cleaned up. The proxy server
// starts a thread to look for clients that are no longer
      // communicating. Use this property to set long a client can
// be idle before it is cleaned up.

      put("com.ibm.as400.access.TunnelProxyServer.clientLifetime", "2700");

      /*---------------------------------------------------------*/
      /* Trace system properties */
      /*---------------------------------------------------------*/

      // This system property specifies which trace categories to enable.
      // This is a comma-delimited list containing any combination of trace
      // categories.The complete list of trace categories is defined in
// the Trace class.
      put ("com.ibm.as400.access.Trace.category", "error,warning,information");

      // This system property specifies the file to which trace output
      // is written.  The default is to write trace output to System.out.
      put ("com.ibm.as400.access.Trace.file", "c:\temp\trace.out");


      /*---------------------------------------------------------*/
      /* Command Call system properties*/
      /*---------------------------------------------------------*/

      // This system property specifies whether CommandCalls should
      // be assumed to be thread-safe. If true, all CommandCalls are
// assumed to be thread-safe. If false, all CommandCalls are
// assumed to be non-thread-safe. This property is ignored
      // for a given CommandCall object if either
```

```
      // CommandCall.setThreadSafe(true/false) or
      // AS400.setMustUseSockets(true) has been performed on the object.
      put ("com.ibm.as400.access.CommandCall.threadSafe", "true");


      /*-------------------------------------------------------*/
      /* Program Call system properties*/
      /*-------------------------------------------------------*/

      // This system property specifies whether ProgramCalls should
      // be assumed to be thread-safe. If true, all ProgramCalls are
// assumed to be thread-safe. If false, all ProgramCalls are
// assumed to be non-thread-safe. This property is ignored
      // for a given ProgramCall object if either
      // ProgramCall.setThreadSafe(true/false) or
      // AS400.setMustUseSockets(true) has been performed on the object.
      put ("com.ibm.as400.access.ProgramCall.threadSafe", "true");


      /*-------------------------------------------------------*/
      /* FTP system properties                                 */
      /*-------------------------------------------------------*/

      // This system property specifies whether the socket is reused
      // for multiple file transfers (through a single FTP instance),
      // when in "active" mode. If true, the socket is reused.
      // If false, a new socket is created for each file transfer.
      // This property is ignored for a given FTP object if
      // FTP.setReuseSocket(true/false) has been performed on the object.
      put ("com.ibm.as400.access.FTP.reuseSocket", "true");


      /*-------------------------------------------------------*/
      /* Connection system properties                          */
      /*-------------------------------------------------------*/

      // This system property specifies the default signon handler.
      // This property is ignored for a given AS400 object if
      // AS400.setSignonHandler() has been performed on
      // the object, or if AS400.setDefaultSignonHandler()
      // has been called.
      put ("com.ibm.as400.access.AS400.signonHandler", "mypackage.MyHandler");

   }
}
```

IBM Toolbox for Java 類別

IBM Toolbox for Java 類別已分類成套件 (如同所有 Java 類別)。每個套件提供某種功能。爲了方便起見，此說明文件通常使用簡短名稱 以參照每個套件。例如，com.ibm.as400.access 套件稱爲存取套件。

請使用下列清單中的鏈結，尋找不同 IBM Toolbox for Java 套件中各類別的相關資訊：

- Access 類別可讓您存取及管理 iSeries 上的資源
- 第 164 頁的『Commtrace 類別』可讓您使用乙太網路或記號環線路說明的通訊追蹤資料
- HTML 類別可讓您快速建立 HTML 套表及表格
- Micro 類別可讓您建立 Java 程式，讓無線裝置直接存取 iSeries 伺服器資料及服務程式。
- ReportWriter 類別可讓您自 XML 資料來源建立格式化文件
- Resource 類別使用一般組織架構來存取及管理 iSeries 資源
- Security 類別建立與伺服器的安全連線，並驗證在 iSeries 伺服器上工作的使用者身分

- Servlet 類別協助擷取及格式化資料以供在 Java Servlet 中使用
- Utility 類別可讓您執行管理作業，例如使用 AS400JarMaker 類別
- Vaccess 類別可讓您以目視方式呈現及操作資料

Access 類別

這些類別可用來存取伺服器的資源。

IBM Toolbox for Java Access 類別代表 iSeries 資料及資源。類別使用 iSeries 伺服器來提供已啓用網際網路的介面，供存取及更新伺服器資料及資源之用。

- AS400 - 管理登入資訊、建立與維護 socket 連線、傳送與接收資料
- SecureAS400 - 可讓您在傳送或接收加密資料時使用 AS400 物件
- AS400JPing - 可讓 Java 程式查詢主電腦伺服器，瞭解哪些服務程式正在執行，以及哪幾個埠正在使用中
- 第 27 頁的『BidiConversionProperties』 - 提供一組可用於控制字集資料轉換的內容
- BidiTransform - 可讓您任意轉換雙向文字
- 第 27 頁的『CallStackEntry』 - 代表伺服器工作特定緒之呼叫堆疊中的登錄
- 叢集雜湊表類別 - 可讓 Java 程式在高度可用的叢集雜湊表中的節點之間，共用及抄寫非持續性資料
- 指令呼叫 - 執行 iSeries 批次指令
- 連線儲存區 - 管理 AS400 物件的儲存區，可用於共用連線及管理使用者所可以擁有的 iSeries 伺服器連線數量
- 資料區 - 建立、存取與刪除資料區
- 資料轉換與說明 - 轉換與處理資料，以及描述資料緩衝區的記錄格式
- 資料佇列 - 建立、存取、變更及刪除資料佇列
- 數位憑證 - 管理 iSeries 伺服器上的數位憑證
- 環境變數 - 管理 iSeries 環境變數
- 事件日誌 - 提供一個方法來記錄異常情況及訊息，此與用來顯示這些異常情況及訊息的裝置無關
- 異常情況 - 例如，當發生裝置錯誤或程式設計錯誤時丟棄錯誤
- FTP - 提供您使用 FTP 功能的可程式化介面
- 整合檔案系統 - 存取檔案、開啓檔案、開啓輸入與輸出串流，及列出目錄內容
- Java 應用程式呼叫 - 呼叫執行於 iSeries Java 虛擬機器上之 iSeries 伺服器的 Java 程式
- JDBC - 存取 DB2® UDB for iSeries 資料
- 工作 - 存取 iSeries 工作及工作日誌
- 訊息 - 存取 iSeries 伺服器上的訊息及訊息佇列
- NetServer 配置 - 存取及修改 iSeries NetServer 的狀態及配置
- 許可權 - 顯示及變更 iSeries 伺服器上的物件權限
- 列印 - 操作 iSeries 列印資源
- 產品授權 - 管理 iSeries 產品的授權
- 程式呼叫 - 呼叫 iSeries 程式
- QSYS 物件路徑名稱 - 代表 iSeries 整合檔案系統中的物件
- 記錄層次存取 - 建立、讀取、更新及刪除 iSeries 檔案及成員
- 服務程式呼叫 - 呼叫 iSeries 服務程式
- 第 39 頁的『SaveFile』 - 代表伺服器上的儲存檔

- 系統狀態 - 顯示系統狀態資訊並允許存取系統儲存區資訊
- 系統值 - 擷取及變更系統值與網路屬性
- 第 40 頁的『Subsystem』 - 代表伺服器上的子系統
- 追蹤 (可服務性) - 記載追蹤點與診斷訊息
- 使用者及群組 - 存取 iSeries 使用者及群組
- 使用者空間 - 存取 iSeries 使用者空間

註: IBM Toolbox for Java 提供名爲資源類別的第二組類別，可用來處理 iSeries 物件及清單。資源類別可提供同屬組織架構及一致的程式設計介面，用以處理不同的 iSeries 物件及清單。在您閱讀過存取套件和 resource 套件中各種類別的相關資訊後，即可針對您的應用程式選擇最適合的物件。

伺服器存取點

IBM Toolbox for Java Access 類別具有類似於使用 IBM iSeries Access for Windows API 的功能。不過，安裝 iSeries Access for Windows，不是使用類別的基本要求。

存取類別會使用現有的 iSeries 伺服器，作爲存取點。在 iSeries 中，每一個伺服器均是以個別工作方式執行，並在 socket 連線中傳送及接收資料串流。

圖 1：伺服器存取點

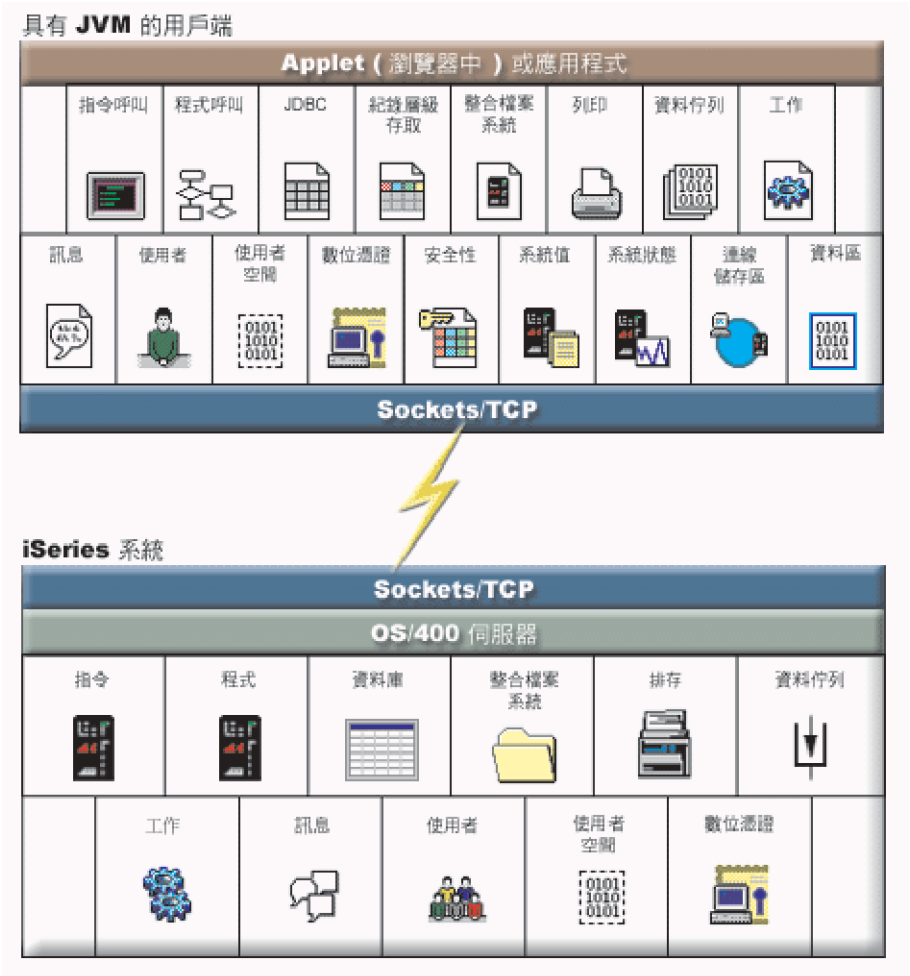


圖 1 的長說明：伺服器存取點 (rzahh501.gif): 位於 IBM Toolbox for Java：伺服器存取點

本圖提供圖形概觀，說明 IBM Toolbox for Java 之存取套件中的類別如何使用 Socket 連線與 iSeries 伺服器上的資料及服務互動。

說明

此圖由下列項目所構成：

- 頂端的矩形代表一或多個用戶端，各有一個 Java 虛擬機器。用戶端的標籤說明該用戶端正在執行瀏覽器中的 Applet 或 Java 應用程式，並備有 Socket/TCP 連線可供 iSeries 伺服器使用。
- 底端的矩形代表 iSeries 伺服器。iSeries 伺服器的標籤說明該系統有一或多部正在執行 i5/OS 的伺服器，而且這些伺服器有 Socket/TCP 連線可供用戶端使用。
- 閃電將兩個矩形連接起來，代表允許資訊在用戶端與 iSeries 伺服器間傳遞的作用中 Socket/TCP 連線。

用戶端 (頂端的矩形) 含有 IBM Toolbox for Java 存取套件中的各種功能，可讓您使用 iSeries 伺服器上的資料及服務。

- 指令呼叫
- 程式呼叫
- JDBC
- 記錄層次存取
- 整合檔案系統
- 列印
- 資料佇列
- 工作
- 訊息
- 使用者
- 使用者空間
- 數位認證
- 安全性
- 系統值
- 系統狀態
- 連線儲存區
- 資料區

iSeries 伺服器 (底端的矩形) 含有不同類型的資料及服務，只要透過 IBM Toolbox for Java 存取套件中的類別，就能加以使用：

- 指令
- 程式
- 資料庫
- 整合檔案系統
- 排存
- 資料佇列
- 工作

- 訊息
- 使用者
- 使用者空間
- 數位認證

AS400 類別

AS400 類別管理一組與伺服器上伺服器工作的 Socket 連線及伺服器的登入行爲，包括提示使用者輸入登入資訊、密碼快取，以及預設使用者管理。

Java 程式若使用某個存取 iSeries 伺服器的類別實例，則 Java 程式必須提供 AS400 物件。例如，CommandCall 物件必須要有 AS400 物件，才能將指令傳送給 iSeries 伺服器。

當 AS400 物件在 iSeries Java 虛擬機器中執行時，會以不同的方式處理連線、使用者 ID 及密碼。如需相關資訊，請參閱 iSeries Java 虛擬機器。

AS400 物件目前支援 Kerberos 鑑別，可透過「Java 一般安全性服務應用程式設計介面 (JGSS API)」來鑑別伺服器，而不是透過使用者 ID 及密碼。

註: 若要使用 Kerberos 通行證，必須先安裝 J2SDK v1.4，並配置「Java 一般安全性服務 (JGSS) 應用程式設計介面」。如需 JGSS 的相關資訊，請參閱 J2SDK, v1.4 Security Documentation  。

如需透過 AS400 物件來管理 iSeries 伺服器連線的相關資訊，請參閱管理連線。請參閱 AS400 連線儲存區，取得如何藉由向連線儲存區要求連線來減少起始連線時間的資訊。

AS400 類別提供下列登入功能：

- 鑑別使用者設定檔
- 取得設定檔記號認證並鑑別相關的使用者設定檔。
- 進行設定檔記號認證的設定
- 管理預設使用者 ID
- 快取密碼
- 使用者 ID 的提示
- 變更密碼
- 取得 iSeries 的版本及版次

如需傳送或接收加密資料時如何使用 AS400 物件的資訊，請參閱 SecureAS400 類別。

管理預設的使用者 ID:

若要將使用者必須登入的次數縮至最小，請使用預設使用者 ID。當 Java 程式未提供使用者 ID 時，該程式會使用預設使用者 ID。您可透過 Java 程式或透過使用者介面來設定預設使用者 ID。如果未建立預設使用者 ID，則「登入」對話視窗可讓使用者設定預設使用者 ID。

一旦對給定的伺服器建立了預設使用者 ID 之後，「登入」對話視窗便不准您改變預設使用者 ID。建構 AS400 物件時，Java 程式可提供使用者 ID 及密碼。程式提供使用者 ID 給 AS400 物件之後，不會影響預設使用者 ID。如果程式要建立或變更預設使用者 ID，則必須將此預設使用者 ID 明確設爲 setUseDefaultUser()。詳細資訊，請參閱提示、預設使用者 ID 與密碼快取摘要。

AS400 物件具有取得、設定及移除預設使用者 ID 的方法。Java 程式也可以經由 setUseDefaultUser() 方法停用預設使用者 ID 處理。如果預設使用者 ID 處理已停用，而且 Java 應用程式未提供使用者 ID，則在每次建立與 iSeries 伺服器的連線時，AS400 物件都會提示使用者輸入使用者 ID。

Java 虛擬機器上代表同一部 iSeries 伺服器的所有 AS400 物件，都使用相同的預設使用者 ID。

在下列範例中，使用兩個 AS400 物件來建立伺服器的兩個連線。當使用者登入時若勾選「Default User ID」方框，則在第二次連線時不會提示輸入使用者 ID。

```
                    // Create two AS400 objects to the
                    // same iSeries.
AS400 sys1 = new AS400("mySystem.myCompany.com");
AS400 sys2 = new AS400("mySystem.myCompany.com");

                    // Start a connection to the command
                    // call service. The user is prompted
                    // for user ID and password.
sys1.connectService(AS400.COMMAND);

                    // Start another connection to the
                    // command call service. The user is
                    // not prompted.
sys2.connectService(AS400.COMMAND);
```

對 iSeries 伺服器的最後一個 AS400 物件進行垃圾收集之後，也會捨棄預設使用者 ID 資訊。

使用密碼快取:

IBM Toolbox for Java 可利用密碼快取功能儲存密碼及使用者 ID 資訊，如此便無需在每一次連線時，提示使用者輸入該項資訊。

您可以使用 AS400 物件提供的方法執行下列作業：

- 清除密碼快取，然後停用密碼快取
- 將使用者必須鍵入登入資訊的次數降至最低

密碼快取適用於 Java 虛擬機器上所有代表 iSeries 伺服器的 AS400 物件。由於 Java 不允許虛擬機器之間的資訊共用，因此各 Java 虛擬機器將看不見彼此所擁有的快取密碼。當將最後一個 AS400 物件棄置垃圾桶之後，便會捨棄此快取。「登入」對話視窗中會提供勾選框，讓使用者選擇是否要快取密碼。在建構 AS400 物件時，Java 程式可讓您選擇提供使用者 ID 及密碼。但不會快取建構子所提供的密碼。

AS400 物件提供清除密碼快取及停用密碼快取的方法。詳細資訊，請參閱提示、預設使用者 ID 與密碼快取摘要。

提示輸入使用者 ID 與密碼:

提示輸入使用者 ID 及密碼可能會在連接至伺服器時產生。提示可由 Java 程式關閉。

Java 程式可以關閉 AS400 物件所顯示的使用者 ID、密碼提示及訊息視窗。當應用程式代表多部用戶端在閘道上執行時，便可能需要執行此動作。如果提示與訊息是顯示在閘道機器，則使用者無法與提示產生互動。這幾種類型的應用程式可以對 AS400 物件執行 setGuiAvailable() 方法而關閉所有的提示。

詳細資訊，請參閱提示、預設使用者 ID 與密碼快取摘要。

提示、預設使用者 ID 與密碼快取摘要:

Java 程式可控制何時提示使用者輸入使用者 ID 及密碼快取。來自「登入」對話框的資訊可用來設定預設使用者 ID 以及快取密碼。下表彙總何時出現提示、 擷取什麼資訊以及設定什麼資訊。

此表假設 Java 程式允許預設使用者 ID 處理及密碼快取，且假設您已勾選「登入」對話框中的**預設使用者 ID** 方框及**儲存密碼**方框。

此表係供您建立用戶端連線，而不是用來在伺服器上執行 Java。

在建構子上提供的系統	在建構子上提供的使用者 ID	在建構子上提供的密碼	已建立預設使用者	使用者 ID 於快取中的密碼	使用已標示之設定值結果
					將提示使用者輸入系統名稱、使用者 ID 及密碼。 已建立預設的使用者 ID 並進行密碼快取。
是					提示使用者輸入使用者 ID 與密碼。顯示系統名稱，但無法變更該名稱。已建立預設的使用者 ID 並進行密碼快取。
是	**是**				提示使用者輸入密碼。顯示使用者 ID，且可變更此 ID。顯示系統名稱，但無法變更該名稱。預設使用者 ID 不會變更。 進行密碼快取。
是	**是**	**是**			無提示。預設使用者 ID 不會變更。不進行密碼快取。
			是		提示輸入系統名稱和密碼。 顯示使用者 ID，且可變更此 ID。變更使用者 ID 不會變更預設使用者 ID。進行密碼快取。
是			**是**		提示輸入預設使用者 ID 的密碼。顯示使用者 ID，且可變更此 ID。顯示系統名稱，但無法變更該名稱。進行密碼快取。

在建構子上提供的系統	在建構子上提供的使用者 ID	在建構子上提供的密碼	已建立預設使用者	使用者 ID 於快取中的密碼	使用已標示之設定值結果
是			是	是	無提示。使用快速記憶體中的預設使用者 ID 及密碼來連線。
是	是			是	無提示。使用快取中的密碼來連接所指定的使用者。
是	是		是	是	無提示。使用快取中的密碼來連接所指定的使用者。
是	是	是	是		無提示。連接成指定的使用者。

SecureAS400 類別

AS400 物件與伺服器通訊時，使用者資料 (使用者密碼除外) 會以未加密形式傳送至伺服器。所以，與 AS400 物件相關聯的 IBM Toolbox for Java 物件，會透過一般的連線與伺服器交換資料。

當您要使用 IBM Toolbox for Java 與伺服器交換機密資料時，可以使用 Secure Sockets Layer (SSL) 將資料加密。請使用 SecureAS400 物件來指定要加密的資料。與 SecureAS400 物件相關聯的 IBM Toolbox for Java 物件，會透過安全連線與伺服器交換資料。

如需相關資訊，請參閱 Secure Sockets Layer 及 Java Secure Socket Extension。

SecureAS400 類別是 AS400 類別的次類別。

您可以經由下列方式，透過建立 SecureAS400 物件的實例，來設定安全伺服器連線：

- SecureAS400(String systemName, String userID) 會提示登入資訊
- SecureAS400(String systemName, String userID, String password) 並不會提示登入資訊

下列範例說明如何使用安全連接，以 CommandCall 傳送指令給 iSeries 伺服器：

```
      // Create a secure AS400 object.This is the only statement that changes
// from the non-SSL case.
      SecureAS400 sys = new SecureAS400("mySystem.myCompany.com");

      // Create a command call object
      CommandCall cmd = new CommandCall(sys, "myCommand");

     // Run the commands.  A secure connection is made when the
     // command is run.  All the information that passes between the
     // client and server is encrypted.
     cmd.run();
```

AS400JPing

AS400JPing 類別可讓 Java 程式查詢主電腦伺服器，瞭解哪些服務程式正在執行，以及哪幾個埠正在使用中。

若要從指令行查詢伺服器，請使用 JPing 類別。

AS400JPing 類別提供許多方法：

- 連通測試 (ping) 伺服器

- 連通測試 (ping) 伺服器上的特定服務程式
- 設定 PrintWriter 物件，此物件爲您所要記載之連通測試 (ping) 資訊
- 爲連通測試 (ping) 作業設定逾時

範例：使用 AS400JPing

下列範例說明如何在 Java 程式內使用 AS400JPing 來連通測試 (ping)「iSeries 遠端指令服務程式」：

```
AS400JPing pingObj = new AS400JPing("myAS400", AS400.COMMAND, false);
  if (pingObj.ping())
    System.out.println("SUCCESS");
  else
    System.out.println("FAILED");
```

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

相關資訊

AS400JPing 類別

BidiTransform 類別

此類別提供佈置轉換功能，可讓您將 iSeries 格式的雙向文字 (在先將它轉換爲 Unicode 之後) 轉換爲 Java 格式的雙向文字，或從 Java 格式轉換爲 iSeries 格式。

AS400BidiTransform 類別

AS400BidiTransform 類別可讓您：

- 取得並設定系統 CCSID
- 取得並設定 iSeries 資料的字串類型
- 取得並設定 Java 資料的字串類型
- 由 Java 佈置轉換資料爲 iSeries
- 由 iSeries 佈置轉換資料爲 Java

範例：使用 AS400BidiTransform

下列範例顯示您如何使用 AS400BidiTransform 類型來轉換雙向文字：

```
// Java data to iSeries layout:
AS400BidiTransform abt;
abt = new AS400BidiTransform(424);
String dst = abt.toAS400Layout("some bidirectional string");
```

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

BidiConversionProperties

BidiConversionProperties 類別提供一組可用於控制字集資料轉換的內容。

相關資訊

BidiConversionProperties Javadoc

CallStackEntry

CallStackEntry 代表伺服器工作特定緒之呼叫堆疊中的登錄。

此類型的物件是藉由呼叫 `Job.getCallStack()` 產生的。

相關資訊

ClusteredHashTable 類別

ClusteredHashTable 類別可讓 Java 程式使用高可用性的叢集雜湊表，以共用及抄寫資料至叢集中之節點間的非持續性儲存體。

若要使用 ClusteredHashTable 類別，請確定您可以讓資料使用非永久性儲存體。抄寫的資料不會加密。

註: 下列資訊假設您瞭解 iSeries 叢集的通用概念及術語。關於叢集及如何使用它們的相關資訊，請參閱叢集。

若要使用 ClusteredHashTable 類別，您必須事先在 iSeries 系統上定義及啓動叢集。您還必須啓動一個有表格伺服器的叢集。詳細資訊，請參閱 Configureclusters 及叢集雜湊表 API。

需要的參數是叢集雜湊表伺服器及 AS400 物件的名稱，此代表了包含叢集雜湊表伺服器的系統。

爲了將資料存放在叢集雜湊表伺服器中，您需要一個連線控點及一個金鑰：

- 當您開啓連線時，叢集雜湊表伺服器會指派供您後續對叢集雜湊表伺服器提出需求時，必須指定的連線控點。此連線控點僅對案例化的 AS400 物件才有用， 所以如果您使用的是不同的 AS400 物件，則需開啓另一個連線。
- 您必須指定金鑰來存取及變更叢集雜湊表格中的資料。不支援重複金鑰。

ClusteredHashTable 類別提供讓您執行下列動作的方法：

- 對叢集雜湊表伺服器工作開啓連線
- 產生唯一的金鑰將資料存入叢集雜湊表
- 對叢集雜湊表伺服器工作關閉使用中連線

有些 ClusteredHashTable 類別中的方法使用 ClusteredHashTableEntry 類別以執行下列動作：

- 自叢集雜湊表取得項目
- 儲存項目於叢集雜湊表中
- 由叢集雜湊表爲所有使用者設定檔取得項目清單

範例：使用 ClusteredHashTable

下列範例操作於名爲 CHTSVR01 的叢集雜湊表伺服器上。它假設有個叢集及叢集雜湊表伺服器已在作用中。它會開啓連線、產生金鑰、使用新金鑰將項目置於叢集雜湊表中、由叢集雜湊表取得項目，並關閉連線。

```
        ClusteredHashTableEntry myEntry = null;

        String myData = new String("This is my data.");
    System.out.println("Data to be stored: " + myData);

    AS400 system = new AS400();

        ClusteredHashTable cht = new ClusteredHashTable(system,"CHTSVR01");

        // Open a connection.
        cht.open();

        // Get a key to the hash table.
        byte[] key = null;
        key = cht.generateKey();

        // Prepare some data that you want to store into the hash table.
        // ENTRY_AUTHORITY_ANY_USER means that any user can access the
        // entry in the clustered hash table.
```

```
// DUPLICATE_KEY_FAIL means that if the specified key already exists,
// the ClusteredHashTable.put() request will not succeed.
int timeToLive = 500;
myEntry = new ClusteredHashTableEntry(key,myData.getBytes(),timeToLive,
          ClusteredHashTableEntry.ENTRY_AUTHORITY_ANY_USER,
          ClusteredHashTableEntry.DUPLICATE_KEY_FAIL);

// Store (or put) the entry into the hash table.
cht.put(myEntry);

// Get an entry from the hash table.
ClusteredHashTableEntry output = cht.get(key);

// Close the connection.
cht.close();
```

使用 ClusteredHashTable 類別會導致 AS400 物件連接伺服器。詳細資訊，請參閱管理連線。

指令呼叫

CommandCall 類別可讓 Java 程式呼叫非互動式 iSeries 指令。

CommandCall 類別

指令的結果可於 AS400Message 物件清單中取得。

CommandCall 的輸入如下：

- 要執行的指令字串
- 代表將執行指令之系統的 AS400 物件

經由 setCommand() 方法，或 run() 方法可以在建構子上設定指令字串。執行指令之後，Java 程式可使用 getMessageList() 方法來擷取由指令產生的任何 iSeries 訊息。

使用 CommandCall 類別，會導致 AS400 物件連接到 iSeries。有關管理連線的資訊，請參閱管理連線。

若 Java 程式及 iSeries 伺服器指令位於相同的伺服器上，根據預設值，IBM Toolbox for Java 行爲會查看系統上指令的緒是否安全。如果安全緒是安全的，則會在緒中執行指令。透過 setThreadSafe() 方法，您可以明確地在指令中指定緒安全，抑制執行時間的查詢。

範例

下列範例所說明的方法，可供您使用 CommandCall 類別來執行不同種類的指令。

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

範例：執行指令

下列範例將說明如何使用 CommandCall 類別在 iSeries 伺服器上執行指令：

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a command call object. This
                    // program sets the command to run
                    // later. It could set it here on the
                    // constructor.
CommandCall cmd = new CommandCall(sys);

                    // Run the CRTLIB command
cmd.run("CRTLIB MYLIB");
```

```
                    // Get the message list which
                    // contains the result of the
                    // command.
    AS400Message[] messageList = cmd.getMessageList();

                    // ... process the message list.

                    // Disconnect since I am done sending
                    // commands to the server
    sys.disconnectService(AS400.COMMAND);
```

範例：執行使用者指定的指令

第 421 頁的『範例：使用 CommandCall』 說明如何執行使用者指定的指令。

連線儲存區

使用連線儲存區可以共用連線及管理 iSeries 伺服器的連線集合 (儲存區)。例如應用程式可以從儲存區擷取連線、使用該連線，再將其傳回儲存區，以供重複使用。

AS400ConnectionPool 類別會管理 AS400 物件的儲存區。AS400JDBCConnectionPool 類別代表 AS400JDBCConnections 的儲存區，Java 程式可以使用其中的連線，這是 IBM Toolbox for Java 支援之 JDBC 2.0 Optional Package API 的一部分。JDBC 3.0 API 也支援 JDBC ConnectionPool 介面，此 API 隨附於 Java 2 Platform 標準版 1.4 版。

不管是哪一種類型的連線儲存區都會保持追蹤它建立的連線數量。使用繼承自 ConnectionPool 的方法可以設定各種連線儲存區內容，包括：

- 一個儲存區可以提供的最大連線數量
- 連線的最大存留時間
- 連線的最大非作用時間

從效能方面來看，連接伺服器是高價的操作。使用連線儲存區可藉由避免重複的連線時間來提升效能。例如，當您藉由將作用中 (已預先連線) 的連線填滿儲存區來建立連線儲存區時建立連線。您可以使用一個連線儲存區輕鬆地擷取、使用、傳回、及重新使用連線物件，而非建立新的連線。

使用 AS400ConnectionPool 來擷取連線，方法是指定系統名稱、使用者 ID、密碼和 (選擇性地) 服務程式。若要指定您要連線的服務程式，請使用 AS400 類別中的常數 (FILE、PRINT，COMMAND 等)。

在擷取及使用連線之後，應用程式會將連線傳回儲存區。將連線傳回儲存區以利重新使用是每一個應用程式的責任。如果沒有將連線傳回儲存區，則連線儲存區的大小會繼續成長且無法重新使用連線。

如需使用 AS400ConnectionPool 類別來開啓 iSeries 連線時，管理連線的相關資訊，請參閱管理連線。

範例：使用 AS400ConnectionPool

第 422 頁的『範例：使用 AS400ConnectionPool』 說明 如何重覆使用 AS400 物件。

資料區

DataArea 類別是一種抽象基礎類別，它代表 iSeries 資料區物件

DataArea

這個基礎類別具有四種次類別來支援：字元資料、小數資料、邏輯資料與含有字元資料的區域資料區。

使用 DataArea 類別，您可以執行下列：

- 取得資料區的大小
- 取得資料區的名稱
- 傳回資料區的 AS400 系統物件
- 重新整理資料區的屬性
- 設定資料區所在的系統

使用 DataArea 類別會導致 AS400 物件連接伺服器。有關管理連線的資訊，請參閱管理連線。

CharacterDataArea

CharacterDataArea 類別代表伺服器上含有字元資料的資料區。字元資料沒有以適當的 CCSID 標示資料的機能；因此資料區物件假設資料在使用者的 CCSID。寫入時，於資料寫入到伺服器之前，資料區物件會從字串 (Unicode) 轉換爲使用者的 CCSID。讀取時，於傳回字串給程式之前，資料區物件會假設資料是使用者的 CCSID，並從 CCSID 轉換爲 Unicode。當讀取資料區的資料時，讀取的資料量是以字元數來計算，而不是以位元組數目來計算。

使用 CharacterDataArea 類別，您可以執行下列：

- 清除資料區，以使它含有所有空白。
- 使用預設內容值，在系統上建立一個字元資料區
- 以特定的屬性建立字元資料區
- 從資料區所在的系統中，刪除資料區
- 傳回資料區所代表的物件整合檔案系統路徑名稱。
- 讀取資料區中含有的全部資料
- 從資料區中位移 0 或您指定的位移讀取資料的指定數量
- 設定資料區的完整整合檔案系統路徑名稱
- 將資料寫入至資料區的開頭
- 寫入資料的指定數量至從位移 0 或您指定之位移開始的資料區

DecimalDataArea

DecimalDataArea 類別代表伺服器上含有小數資料的資料區。

使用 DecimalDataArea 類別，您可以執行下列：

- 清除資料區，使它含有 0.0
- 使用預設內容值，在系統上建立一個小數資料區
- 以特定的屬性建立小數資料區
- 從資料區所在的伺服器中，刪除資料區
- 傳回數字位數至資料區中小數點的右邊
- 傳回資料區所代表的物件整合檔案系統路徑名稱。
- 讀取資料區中含有的全部資料
- 設定資料區的完整整合檔案系統路徑名稱
- 將資料寫入至資料區的開頭

範例：使用 DecimalDataArea 下列範例說明如何建立及寫入小數資料區：

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
      // Establish a connection to the server "My400".
AS400 system = new AS400("MyServer");
      // Create a DecimalDataArea object.
QSYSObjectPathName path = new QSYSObjectPathName("MYLIB", "MYDATA", "DTAARA");
DecimalDataArea dataArea = new DecimalDataArea(system, path.getPath());
      // Create the decimal data area on the server using default values.
dataArea.create();
      // Clear the data area.
dataArea.clear();
      // Write to the data area.
dataArea.write(new BigDecimal("1.2"));
      // Read from the data area.
BigDecimal data = dataArea.read();
      // Delete the data area from the server.
dataArea.delete();
```

LocalDataArea

LocalDataArea 類別代表伺服器上的區域資料區。區域資料區雖是以字元資料資料區的形式存在於伺服器上，但是區域資料區具有一些您需要注意的限制。

區域資料區與伺服器工作相連，所以無法從一另工作中存取它。因此，您無法建立或刪除區域資料區。當伺服器工作結束時，與伺服器工作相連的區域資料區將自動刪除，而且正在參照該工作的 LocalDataArea 物件不再有效。您也應該注意區域資料區在伺服器上具有固定大小：1024 個字元。

使用 LocalDataArea 類別，您可以執行下列：

- 清除資料區，以使它含有所有空白
- 讀取資料區中含有的全部資料
- 從您指定之位移開始的資料區，讀取資料的指定數量
- 將資料寫入至資料區的開頭
- 寫入資料的指定數量至要寫入第一個字元來位移的資料區中

LogicalDataArea

LogicalDataArea 類別代表伺服器上含有邏輯資料的資料區

使用 LogicalDataArea 類別，您可以執行下列：

- 清除資料區，以使之包含 false
- 使用預設內容值，在伺服器上建立一個字元資料區
- 以指定的屬性建立字元資料區
- 從資料區所在的伺服器中，刪除資料區
- 傳回資料區所代表的物件整合檔案系統路徑名稱。
- 讀取資料區中含有的全部資料
- 設定資料區的完整整合檔案系統路徑名稱
- 將資料寫入至資料區的開頭

DataAreaEvent

DataAreaEvent 類別代表資料區事件。

DataAreaEvent 類別可以與任一 DataArea 類別搭配使用。使用 DataAreaEvent 類別，您可以執行下列：

- 取得事件的 ID

DataAreaListener

DataAreaListener 類別提供接收資料區事件的介面。

DataAreaListener 類別可以與任一 DataArea 類別搭配使用。當執行下列任一項時， 您可以呼叫 DataAreaListener 類別：

- 清除
- 建立
- 刪除
- 讀取
- 寫入

資料轉換與說明

資料轉換類別會提供在 iSeries 與 Java 格式之間轉換數值及字元資料的功能。從 Java 程式存取 iSeries 資料時，可能需要轉換。資料轉換類別支援不同數值格式的轉換，及不同 EBCDIC 字碼頁與 Unicode 之間的轉換。

資料說明類別會建置在資料轉換類別中，以便能夠透過單一方法呼叫，來轉換記錄中的所有欄位。RecordFormat 類別可讓程式說明構成下列項目的資料：DataQueueEntry、ProgramCall 參數、透過記錄層次存取類別存取之資料庫檔案中的記錄，或 iSeries 資料的任何緩衝區。「記錄」類別容許程式轉換記錄的內容並依欄位名稱或索引存取資料。

轉換器類別可提供 Java 與 iSeries 伺服器之間快速又有效的轉換。BinaryConverter 可在 Java 位元組陣列與 Java 簡式類型之間轉換。CharConverter 可在 Java String 物件與 i5/OS 字碼頁之間轉換。詳細資訊如下：

Converter

資料類型

AS400DataType 是一個介面，定義資料轉換時所需要的方法。當需要轉換個別片段的資料時，Java 程式將使用資料類型。下列資料類型有轉換類別：

- 數字
- 文字 (字元)
- 組合 (數字與文字)

範例：使用 AS400DataType 類別

下列範例將說明，如何搭配使用 AS400DataType 類別與 ProgramCall，以提供資料給程式參數，及解譯傳回到程式參數的資料。

範例：搭配使用 AS400DataType 類別與 ProgramCall

指定記錄格式的轉換

IBM Toolbox for Java 會提供依據資料類型類別而建置的類別，以逐記錄而非逐欄位的方式，來處理資料的轉換。例如，假設 Java 程式從資料佇列中讀取資料。資料佇列物件即會傳回 iSeries 資料的位元組陣列給 Java 程式。此陣列可能含有多種類型的 iSeries 資料。應用程式可以使用資料類型類別，從位元組陣列中一次轉換一個欄位，或是程式可以建立一個記錄格式，指出以位元組陣列表示的欄位。然後，該記錄即會執行轉換。

當您使用來自程式呼叫、資料佇列與記錄層次存取類別的資料時，記錄格式轉換將非常有用。來自這些類別的輸入與輸出即是可含有多種不同類型的欄位的位元組陣列。透過記錄格式轉換器，可更輕易地在 iSeries 格式與 Java 格式之間轉換此資料。

透過記錄格式的轉換將使用下列三種類別：

- FieldDescription 類別以資料類型與名稱來識別欄位或參數。
- RecordFormat 類別描述一組欄位。
- Record 類別結合了記錄說明 (在 RecordFormat 類別中) 與實際資料。
- LineDataRecordWriter 類別會以行式資料格式將記錄寫入 OutputStream

範例：使用記錄格式轉換類別

下列範例將說明如何搭配使用記錄格式轉換類別與資料佇列：

使用 Record 和 RecordFormat 類別將資料放置於佇列中

使用 FieldDescription、RecordFormat 和 Record 類別

數值資料轉換類別:

數值資料的轉換類別，可將數值資料從 iSeries 伺服器上使用的格式 (在下表中稱爲**伺服器格式**)，轉換成 Java 格式。

支援的類型顯示在下表中：

數值類型	說明
AS400Bin2	在伺服器格式的帶正負號二位元組數字與 Java Short 物件之間轉換。
AS400Bin4	在伺服器格式的帶正負號四位元組數字與 Java Integer 物件之間轉換。
AS400ByteArray	在兩個位元組陣列之間轉換。這是很有用的，因爲轉換器會正確地以零填入並填補目標緩衝區。
AS400Float4	在伺服器格式的帶正負號四位元組浮點數字與 Java Float 物件之間轉換。
AS400Float8	在伺服器格式的帶正負號八位元組浮點數字與 Java Double 物件之間轉換。
AS400PackedDecimal	在伺服器格式的壓縮十進位數與 Java BigDecimal 物件之間轉換。
AS400UnsignedBin2	在伺服器格式的無正負號二位元組數字與 Java Integer 物件之間轉換。
AS400UnsignedBin4	在伺服器格式的無正負號四位元組數字與 Java Long 物件之間轉換。
AS400ZonedDecimal	在伺服器格式的區化十進位數與 Java BigDecimal 物件之間轉換。

範例

下列範例說明使用數值類型的資料如何在伺服器格式與 Java int 之間進行轉換：

範例：從伺服器格式轉換成 Java int

```
// Create a buffer to hold the server data type. Assume the buffer is
// filled with numeric data in the server format by data queues,
// program call, and so on.
     byte[] data = new byte[100];

// Create a converter for this server data type.
        AS400Bin4 bin4Converter = new AS400Bin4();

// Convert from server type to Java object. The number starts at the
// beginning of the buffer.
Integer intObject = (Integer) bin4Converter.toObject(data,0);

// Extract the simple Java type from the Java object.
int i = intObject.intValue();
```

範例：從 Java int 轉換成伺服器格式

```
// Create a Java object that contains the value to convert.
Integer intObject = new Integer(22);

// Create a converter for the server data type.
AS400Bin4 bin4Converter = new AS400Bin4();

// Convert from Java object to server data type.
byte[] data = bin4Converter.toBytes(intObject);

// Find out how many bytes of the buffer were filled with the
// server value.
int length = bin4Converter.getByteLength();
```

文字轉換:

字元資料會經由 AS400Text 類別轉換。此類別可在 EBCDIC 字碼頁及字集 (CCSID) 與 Unicode 之間轉換字元。

AS400Text

當建構 AS400Text 物件時，Java 程式會指定要轉換的字串長度及伺服器 CCSID 或編碼方式。假設 Java 程式的 CCSID 是 13488 Unicode。toBytes() 方法會從 Java 格式轉換成 iSeries 格式的位元組陣列。toObject() 方法則會從 iSeries 格式的位元組陣列轉換爲 Java 格式。

AS400BidiTransform 類別提供佈置轉換功能，可讓您將 iSeries 格式的雙向文字 (在將它轉換成 Unicode 之後) 轉換成 Java 格式的雙向文字，或從 Java 格式轉換成 iSeries 格式。預設的轉換會因工作的 CCSID 而定。若要改變文字的方向及形狀， 請指定 BidiStringType。請注意，在 IBM Toolbox for Java 物件執行內部轉換的位置 (如在 DataArea 類別中)，會有一個置換字串類型的方法。例如，DataArea 類別具有 addVetoableChangeListener() 方法，其可指定來監聽特定內容的拒絕變更，包括字串類型。

範例：轉換文字資料

下列範例假設 DataQueueEntry 物件會以 EBCDIC 方式傳回文字。在此範例中，會將 EBCDIC 資料轉換成 Unicode，使 Java 程式可以使用資料：

```
// Assume the data queue work has already been done to
// retrieve the text from the iSeries and the data has been
// put in the following buffer.
```

```
int textLength = 100;
byte[] data = new byte[textLength];

// Create a converter for the iSeries data type. Note a default
// converter is being built.  This converter assumes the iSeries
// EBCDIC code page matches the client's locale.  If this is not
// true the Java program can explicitly specify the EBCDIC
// CCSID to use. However, it is recommended that you specify a
// CCSID whenever possible (see the Notes: below).
AS400Text textConverter = new AS400Text(textLength)

// Note: Optionally, you can create a converter for a specific
// CCSID. Use an AS400 object in case the program is running
// as an IBM Toolbox for Java proxy client.
int ccsid = 37;
AS400 system = ...; // AS400 object
AS400Text textConverter = new AS400Text(textLength, ccsid, system);

// Note: You can also create a converter with just the AS400 object.
// This converter assumes the iSeries code page matches
// the CCSID returned by the AS400 object.
AS400Text textConverter = new AS400Text(textLength, system);

// Convert the data from EBCDIC to Unicode. If the length of
// the AS400Text object is longer than the number of
// converted characters, the resulting String will be
// blank-padded out to the specified length.
String javaText = (String) textConverter.toObject(data);
```

複合類型轉換類別:

組合類型的轉換類別如下：

- AS400Array - 可讓 Java 程式使用資料類型的陣列。
- AS400Structure - 可讓 Java 程式使用元素爲資料類型的結構。

範例：轉換複合資料類型

下列範例說明 Java 結構與位元組陣列之間的轉換。範例中假定對傳送資料與接收資料使用相同的資料格式。

```
// Create a structure of data types that corresponds to a structure
// that contains: - a four-byte number
//                - four bytes of pad
//                - an eight-byte number
//                - 40 characters
AS400DataType[] myStruct =
{
   new AS400Bin4(),
   new AS400ByteArray(4),
   new AS400Float8(),
   new AS400Text(40)
};

// Create a conversion object using the structure.
AS400Structure myConverter = new AS400Structure(myStruct);

// Create the Java object that holds the data to send to the server.
Object[] myData =
{
   new Integer(88),        // the four-byte number
   new byte[0],            // the pad (let the conversion object 0 pad)
   new Double(23.45),      // the eight-byte floating point number
   "This is my structure"  // the character string
};

// Convert from Java object to byte array.
```

```
byte[] myAS400Data = myConverter.toBytes(myData);


// ... send the byte array to the server. Get data back from the
// server.  The returned data will also be a byte array.


// Convert the returned data from iSeries to Java format.
Object[] myRoundTripData = (Object[])myConverter.toObject(myAS400Data,0);

// Pull the third object out of the structure. This is the double.
Double doubleObject = (Double) myRoundTripData[2];

// Extract the simple Java type from the Java object.
double d = doubleObject.doubleValue();
```

欄位說明類別:

欄位說明類別可讓 Java 程式以資料類型及含有欄位名稱的字串，來說明欄位或參數的內容。如果程式使用記錄層次存取的資料，也就能指定任何可進一步說明欄位的 iSeries 資料定義規格 (DDS) 關鍵字。

欄位說明類別

下列是欄位說明類別：

- BinaryFieldDescription
- CharacterFieldDescription
- DateFieldDescription
- DBCSEitherFieldDescription
- DBCSGraphicFieldDescription
- DBCSOnlyFieldDescription
- DBCSOpenFieldDescription
- FloatFieldDescription
- HexFieldDescription
- PackedDecimalFieldDescription
- TimeFieldDescription
- TimestampFieldDescription
- ZonedDecimalFieldDescription

範例：建立欄位說明

下列範例假設資料佇列中的項目使用相同的格式。每個項目都有訊息編號 (AS400Bin4)、時間戳記 (8 個字元) 以及您可以使用欄位說明來敘述的訊息本文 (50 個字元)：

```
// Create a field description for the numeric data. Note it uses the
// AS400Bin4 data type. It also names the field so it can be accessed by
// name in the record class.
BinaryFieldDescription bfd = new BinaryFieldDescription(new AS400Bin4(), "msgNumber");

// Create a field description for the character data. Note it uses the
// AS400Text data type. It also names the field so it can be accessed by
// name by the record class.
CharacterFieldDescription cfd1 = new CharacterFieldDescription(new AS400Text(8), "msgTime");

// Create a field description for the character data. Note it uses the
```

```
// AS400Text data type. It also names the field so it can be accessed by
// name by the record class.
CharacterFieldDescription cfd2 = new CharacterFieldDescription(new AS400Text(50), "msgText");
```

現在您可以在某個 RecordFormat 類別案例中進行欄位說明的分組。若要瞭解如何將欄位說明新增至 RecordFormat 物件，請參閱下一頁的範例：

『RecordFormat 類別』

RecordFormat 類別:

RecordFormat 類別可讓 Java 程式說明欄位或參數的群組。記錄物件含有 RecordFormat 物件所描述的資料。如果程式將使用記錄層次存取類別，則 RecordFormat 類別亦會容許程式指定索引欄位的說明。

RecordFormat

RecordFormat 物件含有一組欄位說明。欄位說明可透過索引或名稱來存取。方法將存在於 RecordFormat 類別中，以進行下列動作：

- 將欄位說明新增至記錄格式。
- 新增鍵值欄位說明至記錄格式。
- 依照索引或名稱，自記錄格式中擷取欄位說明。
- 依索引或依名稱，自記錄格式中擷取鍵值欄位說明。
- 擷取組成記錄格式的欄位名稱。
- 擷取組成記錄格式之鍵值欄位的名稱。
- 於記錄格式中擷取欄位數。
- 於記錄格式中擷取鍵值欄位數。
- 依照此記錄格式，建立記錄物件。

範例：將欄位說明新增至記錄格式中

在下列範例中，會將欄位說明範例中建立的欄位說明，新增至記錄格式中：

```
// Create a record format object, then fill it with field descriptions.
     RecordFormat rf = new RecordFormat();
     rf.addFieldDescription(bfd);
     rf.addFieldDescription(cfd1);
     rf.addFieldDescription(cfd2);
```

若要瞭解如何從記錄格式中建立記錄，請參閱下一頁的範例：

『Record 類別』

Record 類別:

記錄類別可讓 Java 程式處理記錄格式類別所說明的資料。

Record 類別

資料會在含有伺服器資料及 Java 物件的位元組陣列間轉換。將在記錄類別中提供方法，來執行下列動作：

- 依據索引或名稱擷取欄位的內容，作爲 Java 物件。
- 記錄中擷取欄位數。
- 依據索引或名稱，透過 Java 物件設定欄位的內容。

- 將記錄內容當成伺服器資料，擷取到位元組陣列或輸出串流。
- 從位元組陣列或輸入串流，設定記錄的內容。
- 將記錄內容轉換爲字串。

範例：讀取記錄

下列範例將使用在記錄格式範例中建立的記錄格式：

```
// Assume data queue setup work has already been done. Now read a
// record from the data queue.
DataQueueEntry dqe = dq.read();

// The data from the data queue is now in a data queue entry. Get
// the data out of the data queue entry and put it in the record.
// We obtain a default record from the record format object and
// initialize it with the data from the data queue entry.
Record dqRecord = rf.getNewRecord(dqe.getData());

// Now that the data is in the record, pull the data out one
// field at a time, converting the data as it is removed. The result
// is data in a Java object that the program can now process.
Integer msgNumber = (Integer) dqRecord.getField("msgNumber");
String  msgTime   = (String)  dqRecord.getField("msgTime");
String  msgText   = (String)  dqRecord.getField("msgText");
```

擷取欄位的內容:

您可以使用 Java 程式逐一取得欄位，或一次取得所有欄位，以擷取 Record 物件的內容。

使用 getField() 方法，依照名稱或索引來擷取單一欄位。使用 getFields() 方法， 擷取所有欄位作爲 Object[]。

Java 程式必須強制轉型已傳回給擷取欄位之適當 Java 物件的「物件」(或 Object[] 的元素)。下表會依據欄位類型顯示將強制轉型的適當 Java 物件。

欄位類型 (DDS)	欄位類型 (FieldDescription)	Java 物件
BINARY (B)，長度<= 4	BinaryFieldDescription	短
BINARY (B)，長度 >= 5	BinaryFieldDescription	整數
CHARACTER (A)	CharacterFieldDescription	字串
DBCS Either (E)	DBCSEitherFieldDescription	字串
DBCS Graphic (G)	DBCSGraphicFieldDescription	字串
DBCS Only (J)	DBCSOnlyFieldDescription	字串
DBCS Open (O)	DBCSOpenFieldDescription	字串
DATE (L)	DateFieldDescription	字串
FLOAT (F)，單一精準度	FloatFieldDescription	浮點數
FLOAT (F)，雙倍精度	FloatFieldDescription	倍精確浮點數
HEXADECIMAL (H)	HexFieldDescription	byte[]
PACKED DECIMAL (P)	PackedDecimalFieldDescription	BigDecimal
TIME(T)	TimeDecimalFieldDescription	字串
TIMESTAMP (Z)	TimestampDecimalFieldDescription	字串
ZONED DECIMAL (P)	ZonedDecimalFieldDescription	BigDecimal

SaveFile:

| SaveFile 類別代表伺服器上的儲存檔。

| **相關資訊**

| SaveFile Javadoc

| **Subsystem:**

| Subsystem 類別代表伺服器上的子系統。

| **相關資訊**

| Subsystem Javadoc

設定欄位的內容:

在您的 Java 程式中使用 setField() 方法，藉以設定 Record 物件的內容。

SetField() 方法

Java 程式必須針對將設定的欄位，指定適當的 Java 物件。下表將針對各種可用的欄位類型，列出適當的 Java 物件。

欄位類型 (DDS)	欄位類型 (FieldDescription)	Java 物件
BINARY (B)，長度<= 4	BinaryFieldDescription	短
BINARY (B)，長度 >= 5	BinaryFieldDescription	整數
CHARACTER (A)	CharacterFieldDescription	字串
DBCS Either (E)	DBCSEitherFieldDescription	字串
DBCS Graphic (G)	DBCSGraphicFieldDescription	字串
DBCS Only (J)	DBCSOnlyFieldDescription	字串
DBCS Open (O)	DBCSOpenFieldDescription	字串
DATE (L)	DateFieldDescription	字串
FLOAT (F)，單一精準度	FloatFieldDescription	浮點數
FLOAT (F)，雙倍精度	FloatFieldDescription	倍精確浮點數
HEXADECIMAL (H)	HexFieldDescription	byte[]
PACKED DECIMAL (P)	PackedDecimalFieldDescription	BigDecimal
TIME(T)	TimeDecimalFieldDescription	字串
TIMESTAMP (Z)	TimestampDecimalFieldDescription	字串
ZONED DECIMAL (P)	ZonedDecimalFieldDescription	BigDecimal

LineDataRecordWriter 類別:

LineDataRecordWriter 類別會以行式資料格式將記錄資料寫入 OutputStream 中。 此類別會使用指定的 CCSID 將資料轉譯成位元組。與記錄關聯的記錄格式 會決定資料的格式。

LineDataRecordWriter

使用 LineDataRecordWriter 需要您設定下列記錄格式屬性：

- 記錄格式 ID
- 記錄格式類型

與記錄或 RecordFormat 類別結合使用時，LineDataRecordWriter 會將一筆記錄作爲 writeRecord() 方法的輸入。(當您將記錄案例化時，記錄會以 RecordFormat 爲輸入。)

LineDataRecordWriter 類別提供可讓您執行下列動作的方法：

- 取得 CCSID
- 取得編碼的名稱
- 以行式資料格式將記錄資料寫入 OutputStream

範例：使用 LineDataRecordWriter 類別

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

下列範例說明如何使用 LineDataRecordWriter 類別來撰寫記錄：

```
  // Example using the LineDataRecordWriter class.
 try
{
  // create a ccsid
  ccsid_ = system_.getCcsid();

  // create output queue and specify spooled file data to be *LINE
  OutputQueue outQ = new OutputQueue(system_, "/QSYS.LIB/RLPLIB.LIB/LDRW.OUTQ");
  PrintParameterList parms = new PrintParameterList();
  parms.setParameter(PrintObject.ATTR_PRTDEVTYPE, "*LINE");

  // initialize the record format for writing data
  RecordFormat recfmt = initializeRecordFormat();

  // create a record and assign data to be printed...
  Record record = new Record(recfmt);
  createRecord(record);

  SpooledFileOutputStream os = null;
  try {
      // create the output spooled file to hold the record data
      os = new SpooledFileOutputStream(system_, parms, null, outQ);
  }
  catch (Exception e) {
      System.out.println("Error occurred creating spooled file");
      e.printStackTrace();
  }

  // create the line data record writer
  LineDataRecordWriter ldw;
  ldw = new LineDataRecordWriter(os, ccsid_, system_);

  // write the record of data
  ldw.writeRecord(record);

  // close the outputstream
  os.close();
}

catch (Exception e)
{
  failed(e, "Exception occurred.");
}
```

資料佇列

DataQueue 類別可讓 Java 程式與伺服器資料佇列彼此互動。

iSeries 伺服器上的資料佇列有下列特性：

- 資料佇列容許工作間的快速通訊。因此，它是一種在工作間進行資料同步與傳遞的非常好方式。
- 許多工作可同時存取資料佇列。
- 資料佇列上的訊息是沒有格式限制。不同於資料庫檔案，它不需要欄位。
- 資料佇列可用於同步與非同步處理。
- 資料佇列上的訊息可以按下列方式排序：
  - 後進先出 (LIFO)。資料佇列的最後一個 (最新) 訊息是第一個離開佇列的訊息。
  - 先進先出 (FIFO)。資料佇列的第一個 (最舊) 訊息是離開佇列的第一個訊息。
  - 索引式。資料佇列中的每一個訊息都有一個相關的索引。只能藉由指定與某訊息相關的索引才能從佇列中取出該訊息。

資料佇列類別提供了一組完整的介面，可讓您從 Java 程式存取伺服器資料佇列。它是一種在 Java 程式與伺服器上以任何程式設計語言編寫的程式之間進行通訊的好方法。

每一個資料佇列物件的必要參數是 AS400 物件，代表具有資料佇列或要建立資料佇列的伺服器系統。

使用資料佇列類別會導致 AS400 物件連接伺服器。有關管理連線的資訊，請參閱管理連線。

每一個資料佇列物件都需要資料佇列的整合檔案系統路徑名稱。資料佇列的類型是 DTAQ。請參閱整合檔案系統路徑名稱，取得詳細資訊。

循序與索引資料佇列

資料佇列類別支援下列資料佇列：

- 循序資料佇列
- 索引資料佇列

這兩種佇列類型的共同方法是在 BaseDataQueue 類別中。DataQueue 類別會擴充 BaseDataQueue 類別，以便完成循序資料佇列的實施。BaseDataQueue 類別是由 KeyedDataQueue 類別加以擴充，以完成索引資料佇列的實施。

從某資料佇列讀取資料時，資料會存放在 DataQueueEntry 物件。此物件保留索引式與循序資料佇列的資料。從索引資料佇列中讀取資料時，其它可用的資料是放在擴充 DataQueueEntry 類別的 KeyedDataQueueEntry 物件中。

資料佇列類別不會變更寫入至或讀取自伺服器資料佇列的資料。Java 程式必須正確格式化此資料。資料轉換類別會提供一些方法來轉換資料。

範例：使用 DataQueue 和 DataQueueEntry

下列範例建立 DataQueue 物件，從 DataQueueEntry 物件讀取資料，然後切斷系統。

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
// Create an AS400 object
AS400 sys = new AS400("mySystem.myCompany.com");

// Create the DataQueue object
DataQueue dq = new DataQueue(sys, "/QSYS.LIB/MYLIB.LIB/MYQUEUE.DTAQ");

// read data from the queue
DataQueueEntry dqData = dq.read();

// get the data out of the DataQueueEntry object.
```

```
byte[] data = dqData.getData();

// ... process the data

// Disconnect since I am done using data queues
sys.disconnectService(AS400.DATAQUEUE);
```

循序資料佇列:

依先進先出 (FIFO) 或後進先出 (LIFO) 順序來除去伺服器上循序資料佇列中的登錄。

BaseDataQueue 及 DataQueue 類別提供下列方法，以使用循序資料佇列：

- 在伺服器上建立一個資料佇列。Java 程式必須指定資料佇列中的最大登錄大小。建立此佇列時，Java 程式可選擇性地指定其他資料佇列參數 (FIFO 或 LIFO、儲存傳送者資訊、指定權限資訊、強制寫至磁碟，以及提供佇列說明)。
- 察看資料佇列中的登錄，但不從佇列中除去登錄。如果目前在佇列中沒有登錄，則 Java 程式可等待或立即返回。
- 從佇列中讀出登錄。如果佇列中沒有登錄，則 Java 程式可等待或立即返回。
- 將登錄寫入到佇列。
- 從佇列清除所有登錄。
- 刪除佇列。

BaseDataQueue 類別會提供其它方法來擷取資料佇列的屬性。

範例：使用循序資料佇列

下列循序資料佇列範例，將說明生產者如何將項目放置於資料佇列，以及消費者如何從佇列中取出項目加以處理：

第 427 頁的『範例：使用 DataQueue 類別以在佇列上放置資料』

第 430 頁的『範例：使用 DataQueue 類別以讀取資料佇列登錄』

索引資料佇列:

BaseDataQueue 及 KeyedDataQueue 類別提供使用索引資料佇列的方法。

BaseDataQueue

KeyedDataQueue

- 在伺服器上建立一個索引資料佇列。Java 程式必須指定佇列中的索引長度及最大登錄大小。Java 程式可選擇性地指定權限資訊、儲存傳送者資訊、強制寫至磁碟，以及提供佇列說明。
- 依照指定的金鑰察看某一登錄，而不將它從佇列中移除。如果目前在佇列中沒有符合索引準則的登錄，則 Java 程式可等待或立即返回。
- 依照指定的金鑰，讀取佇列中的登錄。如果在佇列中沒有符合索引準則的登錄，則 Java 程式可等待或立即返回。
- 寫入索引 登錄到佇列中。
- 清除所有登錄或所有符合指定金鑰的登錄。
- 刪除佇列。

BaseDataQueue 和 KeyedDataQueue 類別也提供其它方法來擷取資料佇列的屬性。

範例：使用索引資料佇列

下列索引資料佇列範例，將說明生產者如何將項目放置於資料佇列，以及消費者如何從佇列中取出項目加以處理：

第 435 頁的『範例：使用 KeyedDataQueue』

第 438 頁的『範例：使用 KeyedDataQueue 類別以讀取資料佇列登錄』

數位認證

數位認證是以數字簽署的聲明，用於保護透過網際網路的交易。

數位憑證可在執行 i5/OS 版本 4 版次 3 (V4R3) 與以上版本的伺服器中使用。 若要使用 Secure Sockets Layer (SSL) 以使連線安全，將需要數位認證。

數位認證是由下列所構成：

- 使用者的公用暗碼鍵
- 使用者的名稱與位址
- 協力廠商資格認定權限 (CA) 的數位簽名。權限的簽名表示使用者是可靠的實體。
- 認證的發出日期
- 認證的有效日期

作爲安全伺服器的管理員，您可以對伺服器新增資格認定權限的「授信單位鍵」。這表示您的伺服器將信任已透過該特殊資格認定權限所認證的任何人員。

數位認證也會提供暗碼，透過專用暗碼鍵確保資料轉送的安全。

您可以透過 javakey 工具來建立數位認證。(如需 javakey 及 Java 安全性的相關資訊，請參閱 Sun Microsystems, Inc., Java Security 網頁 。) IBM Toolbox for Java 授權程式含有一些可在 iSeries 伺服器上管理數位憑證的類別。

AS400Certificate 類別會提供不同方式來管理 X.509 ASN.1 編碼憑證。提供的類別會執行下列事項：

- 取得與設定憑證資料。
- 依驗證清單或使用者設定檔列示憑證。
- 管理憑證，例如，對使用者設定檔新增憑證，或從驗證清單中刪除憑證。

使用憑證類別將導致 AS400 物件連線至伺服器。有關管理連線的資訊，請參閱管理連線。

在伺服器上，憑證屬於驗證清單或使用者設定檔。

- AS400CertificateUserProfileUtil 類別具有管理使用者設定檔中憑證的方法。
- AS400CertificateVldlUtil 類別具有管理驗證清單中憑證的方法。

若要使用 AS400CertificateUserProfileUtil 和 AS400CertificateVldlUtil，必須先安裝基本作業系統選項 34 (數位憑證管理程式)。這兩種類別擴充了 AS400CertificateUtil，這是一個摘要基礎類別，定義這兩種次類別共用的方法。

AS400Certificate 類別提供讀取及寫入憑證資料的方法。資料是以位元組陣列方式存取。Java 虛擬機器 1.2 中的 Java.Security 套件提供可以用來取得及設定憑證個別欄位的類別。

列示憑證

若要取得憑證的清單，Java 程式必須執行下列動作：

1. 建立 AS400 物件。
2. 建構正確的憑證物件。不同物件係針對使用者設定檔中的列示驗證 (AS400CertificateUserProfileUtil) 及驗證清單中的列示憑證 (AS400CertificateVldlUtil) 而使用。
3. 依據憑證屬性建立選取準則。AS400CertificateAttribute 類別包含作爲選取準則使用的屬性。一或多個屬性物件會定義在憑證新增到清單中之前必須符合的準則。例如，清單可能僅含有某個使用者或組織的憑證。
4. 在伺服器上建立一個使用者空間，並將憑證置於該使用者空間中。大量資料可經由列示作業來產生。必須先將資料置於使用者空間中，Java 程式才能加以擷取。使用 listCertificates() 方法，將憑證置於使用者空間中。
5. 使用 getCertificates() 方法，自使用者空間擷取憑證。

範例：列出數位憑證

下面範例會列出驗證清單中的憑證。它僅會列示那些屬於某人的憑證。

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
        // Create an AS400 object.
The certificates are on this system.
      AS400 sys = new AS400("mySystem.myCompany.com");

        // Create the certificate object.
     AS400CertificateVldlUtil certificateList =
        new AS400CertificateVldlUtil(sys, "/QSYS.LIB/MYLIB.LIB/CERTLIST.VLDL");

        // Create the certificate attribute list. We only want certificates
        // for a single person so the list consists of only one element.
     AS400CertificateAttribute[] attributeList = new AS400CertificateAttribute[1];
     attributeList[0] =
        new AS400CertificateAttribute(AS400CertificateAttribute.SUBJECT_COMMON_NAME, "Jane Doe");

        // Retrieve the list that matches the criteria. User space "myspace"
        // in library "mylib" will be used for storage of the certificates.
        // The user space must exist before calling this API.
     int count = certificateList.listCertificates(attributeList, "/QSYS.LIB/MYLIB.LIB/MYSPACE.USRSPC");

        // Retrieve the certificates from the user space.
     AS400Certificates[] certificates =
        certificateList.getCertificates("/QSYS.LIB/MYLIB.LIB/MYSPACE.USRSPC", 0, 8);

        // Process the certificates
```

EnvironmentVariable 類別

EnvironmentVariable 類別及 EnvironmentVariableList 類別可讓您存取及設定 iSeries 系統層次的環境變數。

EnvironmentVariable 類別

EnvironmentVariableList 類別

每個變數都有唯一的 ID：系統名稱及環境變數名稱。每個環境變數都會連結一個 CCSID，而且是預設成現行作業的 CCSID，用來說明存放變數內容的位置。

註: 儘管環境變數與系統值的用途相近，不過它們並不相同。請參閱 SystemValues，以取得如何存取系統值的詳細資訊。

使用 EnvironmentVariable 物件以對環境變數執行下列動作：

- 取得及設定名稱
- 取得及設定系統
- 取得及設定值 (可讓您變更 CCSID)
- 重新整理值

範例：建立、設定及取得環境變數

下列範例會建立兩個 EnvironmentVariables，並且設定及取得它們的值。

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
// Create the iSeries system object.
AS400 system = new AS400("mySystem");

// Create the foreground color environment variable and set it to red.
EnvironmentVariable fg = new EnvironmentVariable(system, "FOREGROUND");
fg.setValue("RED");

// Create the background color environment variable and get its value.
EnvironmentVariable bg = new EnvironmentVariable(system, "BACKGROUND");
String background = bg.getValue();
```

異常情況

發生裝置錯誤、實體限制、程式設計錯誤或使用者輸入錯誤時，IBM Toolbox for Java Access 類別會丟出異常。根據發生的錯誤類型而不是出現錯誤的位置來決定異常情況類別。

大部分異常都含有下列資訊：

- **錯誤類型**：丟出的異常物件會指出發生的錯誤類型。同一類型的錯誤會在某異常情況類別中組成群組。
- **錯誤明細**：異常含有回覆碼，可進一步識別錯誤發生的原因。回覆碼值是異常情況類別中的常數。
- **錯誤文字**：異常含有字串，可用來說明發生的錯誤。該字串會以用戶端 Java 虛擬機器的語言環境來轉換。

範例：攔截丟出的異常

下面範例會顯示如何攔截擲出的異常、擷取回覆碼及顯示異常文字：

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
// All the setup work to delete a file on the server through the
// IFSFile class is done. Now try deleting the file.
 try
{
           aFile.delete();

}

// The delete failed.
catch (ExtendedIOException e)
{
   // Display the translated string containing the reason that the
   // delete failed.
   System.out.println(e);

   // Get the return code out of the exception and display additional
   // information based on the return code.
   int rc = e.getReturnCode()

   switch (rc)
   {
```

```
        case ExtendedIOException.FILE_IN_USE:
           System.out.println("Delete failed, file is in use "):
           break;

        case ExtendedIOException.PATH_NOT_FOUND:
           System.out.println("Delete failed, path not found ");
           break;

        // For every specific error that you want to track...

        default:
           System.out.println("Delete failed, rc = ");
           System.out.println(rc);
     }
  }
```

FTP 類別

FTP 類別具有使用 FTP 功能的可程式化介面。

FTP 類別

您不必再於個別的應用程式中使用 java.runtime.exec() 或要求使用者執行 FTP 指令。亦即，您可以直接對您的應用程式進行 FTP 功能的程式設計。所以，在您的程式中，您可以執行下列：

- 連接 FTP 伺服器
- 傳送指令到伺服器
- 列出目錄中 的檔案
- 取得伺服器中的檔案及
- 放入檔案到伺服器內

範例：使用 FTP 從伺服器複製檔案

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

例如，利用 FTP 類別，您即可從伺服器的目錄中複製一組檔案：

```
FTP client = new FTP("myServer", "myUID", "myPWD");
client.cd("/myDir");
     client.setDataTransferType(FTP.BINARY);
     String [] entries = client.ls();

     for (int i = 0; i < entries.length; i++)
{
  System.out.println("Copying " + entries[i]);
   try
  {
    client.get(entries[i], "c:\\ftptest\\" + entries[i]);
  }
  catch (Exception e)
  {
    System.out.println("  copy failed, likely this is a directory");
  }
}

     client.disconnect();
```

FTP 是使用許多不同 FTP 伺服器的同屬介面。因此，程式設計師可以決定伺服器的語意。

AS400FTP 子類別

當 FTP 類別是同屬 FTP 介面時，則會明確地寫入伺服器上 FTP 伺服器的 AS400FTP 子類別。亦即，它瞭解 iSeries 伺服器上 FTP 伺服器的語意。例如，此類別瞭解將儲存檔轉送到伺服器時所需要的各種步驟，必會自動執行這些步驟。AS400FTP 也結合了 IBM Toolbox for Java 的安全機能。在使用其他 IBM Toolbox for Java 類別時，AS400FTP 會視 AS400 物件的不同而要求系統名稱、使用者 ID 及密碼。

範例：使用 AS400FTP 將檔案儲存到伺服器

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

在下列範例中，會將儲存檔放在伺服器上。請注意，應用程式並未將資料轉送類型設爲二進位，或使用 CommandCall 來建立儲存檔。既然副檔名是 .savf，AS400FTP 類別會偵測到要放入的檔案是一個儲存檔，所以它會自動執行這些步驟。

```
AS400 system = new AS400();
 AS400FTP ftp = new AS400FTP(system);
ftp.put("myData.savf", "/QSYS.LIB/MYLIB.LIB/MYDATA.SAVF");
```

整合檔案系統

整合檔案系統類別可讓 Java 程式以位元組串流或字元串流的方式，存取 iSeries 伺服器上的整合檔案系統中的檔案。因爲 java.io 套件沒有提供檔案重新導向及其他 iSeries 功能，所以建立了整合檔案系統類別。

IFSFile 類別提供的功能是由 java.io 套件中的檔案 IO 類別提供的功能超集。java.io FileInputStream、FileOutputStream 以及 RandomAccessFile 內的所有方法都是在整合檔案系統類別之內。

此外，這些類別還包含一些方法可執行下列動作：

- 指定檔案共用模式來拒絕存取使用中的檔案
- 指定檔案建立模式來開啓、建立或置換此檔案
- 鎖定檔案的某區段並在使用該檔案時拒絕該部分的存取
- 更有效率地列示目錄的內容
- 藉由限制呼叫伺服器來快取目錄的內容，以增進效能
- 決定伺服器檔案系統中可用的位元組數
- 讓 Java Applet 存取伺服器檔案系統中的檔案
- 以文字而非二進位資料來讀取與寫入資料
- 決定當物件於 QSYS.LIB 檔案系統中時，該檔案物件的類型 (邏輯、實體、儲存等)。

透過整合檔案系統類別，Java 程式，可直接存取 iSeries 上的串流檔。Java 程式仍可使用 java.io 套件，但用戶端作業系統此時必須提供重新導向的方法。例如，如果 Java 程式正在 Windows 95 或 Windows NT 作業系統上執行，則需要 iSeries Access for Windows 的「網路磁碟機」功能，將 java.io 呼叫重新導向至 iSeries。透過整合檔案系統類別，您就不需要 iSeries Access for Windows。

整合檔案系統類別的必要參數之一是 AS400 物件，該物件代表包含此檔案的 iSeries 系統。使用整合檔案系統類別，會導致 AS400 物件連接到 iSeries。有關管理連線的資訊，請參閱管理連線。

整合檔案系統類別需要此物件在整合檔案系統中的階層式名稱。使用正斜線作爲路徑分隔字元。下面範例將告訴您如何存取目錄路徑 DIR1/DIR2 中的 FILE1：

```
/DIR1/DIR2/FILE1
```

整合檔案系統類別

下面表格列出整合檔案系統類別：

整合檔案系統類別	說明
IFSFile	代表在整合檔案系統中的檔案
IFSJavaFile	代表在整合檔案系統中的檔案 (擴充 java.io.File)
IFSFileInputStream	代表從 iSeries 檔案讀取資料的輸入串流
IFSTextFileInputStream	代表從檔案讀取字元資料的串流 (已棄用)
第 52 頁的『IFSFileReader』	使用此類別讀取整合檔案系統中的字元檔案。
IFSFileOutputStream	代表將資料寫入 iSeries 檔案的輸出串流
第 54 頁的『IFSFileWriter』	使用 IFSFileWriter 寫入整合檔案系統中的字元檔案。
IFSTextFileOutputStream	代表從檔案讀取字元資料的串流 (已棄用)
IFSRandomAccessFile	代表 iSeries 上用來讀取及寫入資料的檔案
IFSFileDialog	容許使用者在檔案系統內移動，以及在檔案系統內選取檔案
第 57 頁的『IFSSystemView』	IFSSystemView 提供 iSeries 整合檔案系統的閘道，用於建構 javax.swing.JFileChooser 物件時。

範例：使用整合檔案系統類別

第 446 頁的『範例：使用 IFS 類別以將檔案從某一目錄複製到另一目錄』說明如何使用整合檔案系統類別，將檔案從 iSeries 的一個目錄複製到另一個目錄上。

第 448 頁的『範例：使用 IFS 範例列出目錄內容』 說明如何使用整合檔案系統類別列出 iSeries 上某個目錄的內容。

IFSFile 類別:

IFSFile 類別代表 iSeries 整合檔案系統中的物件。

IFSFile

IFSFile 的方法代表對整個物件所執行的一些作業。您可使用 IFSFileInputStream、IFSFileOutputStream 以及 IFSRandomAccessFile 來讀取和寫入此檔案。IFSFile 類別可讓 Java 程式執行下列動作：

- 確定物件是否存在，而且是一個目錄或檔案
- 決定 Java 程式是否可以讀取或寫入檔案。
- 確定檔案的長度
- 確定物件的許可權並設定物件的許可權
- 建立目錄
- 刪除檔案或目錄
- 更名檔案或目錄
- 取得或設定檔案的前次修改日期
- 列示目錄內容
- 列出目錄內容並將屬性資訊存入本端快取
- 決定系統上的可用空間量
- 決定當檔案物件位於 QSYS.LIB 檔案系統中時，該檔案物件的類型

您可以使用 list() 方法或 listFiles() 方法，取得目錄中的檔案清單：

- listFiles() 方法會透過初始呼叫快取每個檔案的資訊。在呼叫 listFiles() 後，使用其它方法來查詢檔案明細會有較佳的效能，因爲資料是擷取自快取。例如，呼叫 isDirectory() 取得從 listFiles() 傳回的 IFSFile 物件不需要 呼叫伺服器。
- list() 方法在擷取每個檔案的相關資訊時，必須對伺服器提出不同的要求，因而會使執行速度變慢，並耗用更多伺服器資源。

註: 使用 listFiles() 即表示，快取中的資訊可能已無時效性，因此必須再次呼叫 listFiles() 來重新整理資料。

範例

下列範例說明如何使用 IFSFile 類別：

- 第 441 頁的『範例：建立目錄 』
- 第 442 頁的『範例：使用 IFSFile 異常以追蹤錯誤 』
- 第 442 頁的『範例：列出具有 .txt 副檔名的檔案 』
- 第 443 頁的『範例：使用 IFSFile listFiles() 方法以列出目錄內容』

IFSJavaFile 類別:

此類別代表 iSeries 整合檔案系統中的檔案，並延伸 java.io.File 類別。IFSJavaFile 可讓您寫入 java.io.File 介面的檔案，以存取 iSeries 整合檔案系統。

IFSJavaFile

IFSJavaFile 將產生與 java.io.File 相容的可攜性介面，並僅使用 java.io.File 使用的錯誤與異常情況。IFSJavaFile 會使用來自 java.io.File 的安全管理程式特性， 但不像 java.io.File，IFSJavaFile 會不斷地使用安全特性。

IFSJavaFile 可以搭配 IFSFileInputStream 及 IFSFileOutputStream 一起使用。它不支援 java.io.FileInputStream 與 java.io.FileOutputStream。

IFSJavaFile 是以 IFSFile 爲基礎；它的介面比 IFSFile 更像 java.io.File。IFSFile 是 IFSJavaFile 的替代類別。

您可以使用 list() 方法或 listFiles() 方法，以取得目錄中的檔案清單：

- listFiles() 方法會執行得比較好，因爲它會在起始呼叫中擷取及快取每一個檔案的資訊。之後，則會從快取記憶體中擷取每一個檔案的相關資訊。
- list() 方法會在個別要求中擷取每一個檔案的相關資訊，這使得它的執行速度較慢且需要更多伺服器資源。

註: 使用 listFiles() 即表示，快取中的資訊已無時效性，因此必須重新整理這些資料。

範例：使用 IFSJavaFile

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

下面範例說明 IFSJavaFile 類別的使用方法：

```
// Work with /Dir/File.txt on the system flash.
AS400 as400 = new AS400("flash");
IFSJavaFile file = new IFSJavaFile(as400, "/Dir/File.txt");

// Determine the parent directory of the file.
String directory = file.getParent();

// Determine the name of the file.
String name = file.getName();
```

```
        // Determine the file size.
        long length = file.length();

        // Determine when the file was last modified.
        Date date = new Date(file.lastModified());

        // Delete the file.
        if (file.delete() == false)
      {
           // Display the error code.
           System.err.println("Unable to delete file.");
}

       try
      {
        IFSFileOutputStream os =
           new IFSFileOutputStream(file.getSystem(), file, IFSFileOutputStream.SHARE_ALL, false);
           byte[] data = new byte[256];
           int i = 0;
           for (; i < data.length; i++)
        {
               data[i] = (byte) i;
               os.write(data[i]);
        }
        os.close();
      }
      catch (Exception e)
      {
           System.err.println ("Exception: " + e.getMessage());
      }
```

IFSFileInputStream:

IFSFileInputStream 類別代表從伺服器上檔案讀取資料的輸入串流。

IFSFileInputStream

如同在 IFSFile 類別中，IFSFileInputStream 中的方法從 java.io 套件中複製 FileInputStream 的方法。除了這些方法以外，IFSFileInputStream 還有其他 iSeries 伺服器專用的方法。IFSFileInputStream 類別可讓 Java 程式執行下列動作：

- 開啓檔案以讀取。因爲此類別不會在伺服器上建立檔案，所以檔案必須存在。您可以使用建構子，它可讓您指定檔案共用模式。
- 決定串流中的位元組數。
- 讀取串流中的位元組。
- 略過串流中的位元組。
- 鎖定或解除鎖定串流中的位元組。
- 關閉檔案。

如同在 java.io 中的 FileInputStream，此類別可讓 Java 程式從檔案讀取位元組的串流。Java 只能循序讀取位元組，或是略過串流中的位元組。

除了 IFSFileInputStream 中的方法以外，IFSFileInputStream 還可讓 Java 程式執行下列作業：

- 鎖定和解除鎖定串流的位元組。詳細資訊，請參閱 IFSKey。
- 開啓檔案之後指定共用模式。詳細資訊，請參閱共用模式。

範例：使用 IFSFileInputStream

下面範例說明如何使用 IFSFileInputStream 類別。

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Open a file object that
                    // represents the file.
IFSFileInputStream aFile = new IFSFileInputStream(sys,"/mydir1/mydir2/myfile");

                    // Determine the number of bytes in
                    // the file.
int available = aFile.available();

                    // Allocate a buffer to hold the data
byte[] data = new byte[10240];

                    // Read the entire file 10K at a time
for (int i = 0; i < available; i += 10240)
{
    aFile.read(data);
}

                    // Close the file.
aFile.close();
```

IFSTextFileInputStream 類別:

IFSTextFileInputStream 已棄用，並已置換爲類別 IFSFileReader。

IFSTextFileInputStream 類別代表從檔案讀取的字元資料串流。讀取自 IFSTextFileInputStream 物件的資料，會提供給 Java String 物件中的 Java 程式，因此它一律爲 Unicode。開啓檔案之後，IFSTextFileInputStream 物件會判斷檔案中的資料的 CCSID。如果資料是以 Unicode 以外的編碼儲存的，則 IFSTextFileInputStream 物件會在將資料提供給 Java 程式之前，先將資料從檔案的編碼轉換爲 Unicode。如果無法轉換資料，則會丟出 UnsupportedEncodingException。

下面範例說明如何使用 IFSTextFileInputStream：

```
                    // Work with /File on the system
                    // mySystem.
AS400 as400 = new AS400("mySystem");
IFSTextFileInputStream file = new IFSTextFileInputStream(as400, "/File");

                    // Read the first four characters of
                    // the file.
String s = file.read(4);

                    // Display the characters read. Read
                    // the first four characters of the
                    // file. If necessary, the data is
                    // converted to Unicode by the
                    // IFSTextFileInputStream object.
System.out.println(s);

                    // Close the file.
file.close();
```

相關參考

『IFSFileReader』
使用此類別讀取整合檔案系統中的字元檔案。

IFSFileReader:

使用此類別讀取整合檔案系統中的字元檔案。

IFSFileReader 的用途在於讀取字元的串流。IFSFileReader 會置換 IFSTextFileOutputStream。

範例：使用 IFSFileReader

下列範例會說明 IFSFileReader 的使用方式：

```
import java.io.BufferedReader;

 // Work with /File1 on the system eniac.
AS400 system = new AS400("eniac");
IFSFile file = new IFSFile(system, "/File1");
BufferedReader reader = new BufferedReader(new IFSFileReader(file));

 // Read the first line of the file, converting characters.
String line1 = reader.readLine();

 // Display the String that was read.
System.out.println(line1);

 // Close the reader.
reader.close();
```

相關資訊

IFSFileReader Javadoc

IFSFileOutputStream 類別:

IFSFileOutputStream 類別代表寫入資料到伺服器上之檔案的輸出串流。

IFSFileOutputStream

如同在 IFSFile 類別中，IFSFileOutputStream 中的方法從 java.io 套件中複製 FileOutputStream 的方法。IFSFileOutputStream 也有專供伺服器使用的其它方法。IFSFileOutputStream 類別可讓 Java 程式執行下列動作：

- 開啓檔案以寫入。如果此檔案已存在，則會置換該檔案。您可使用建置器來指定檔案共用模式，以及是否已經附加現有檔案的內容。
- 寫入位元組到串流。
- 確定位元組寫入串流的磁碟。
- 鎖定或解除鎖定串流中的位元組。
- 關閉檔案。

如同在 java.io 中的 FileOutputStream，此類別可讓 Java 程式循序地將位元組串流寫入檔案中。

除了 FileOutputStream 中的方法以外，IFSFileOutputStream 還可讓 Java 程式執行下列作業：

- 鎖定和解除鎖定串流的位元組。詳細資訊，請參閱 IFSKey。
- 開啓檔案之後指定共用模式。詳細資訊，請參閱共用模式。

範例：使用 IFSFileOutputStream

下面範例說明 IFSFileOutputStream 類別的使用方法：

```
                    // Create an AS400 object
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Open a file object that
                    // represents the file.
```

```
IFSFileOutputStream aFile = new IFSFileOutputStream(sys,"/mydir1/mydir2/myfile");

                  // Write to the file
byte i = 123;
aFile.write(i);

                  // Close the file.
aFile.close();
```

IFSTextFileOutputStream 類別:

IFSTextFileOutputStream 已棄用，並已置換爲類別 IFSFileWriter。

IFSTextFileOutputStream 類別代表寫入檔案的字元資料串流。提供給 IFSTextFileOutputStream 物件的資料位於 Java String 物件中，因此輸入一律爲 Unicode。不過當資料寫入檔案時，IFSTextFileOutputStream 物件可轉換此資料爲另一個 CCSID。預設行爲是將 Unicode 字元寫入檔案中，但是 Java 程式可以在開啓檔案之前，先設定目標 CCSID。在此情況下，IFSTextFileOutputStream 物件會先將字元從 Unicode 轉換成指定的 CCSID，然後才將它們寫入檔案。如果無法轉換資料，則會丟出 UnsupportedEncodingException。

範例：使用 IFSTextFileOutputStream

下面範例說明如何使用 IFSTextFileOutputStream：

```
                  // Work with /File on the system
                  // mySystem.
AS400 as400 = new AS400("mySystem");
IFSTextFileOutputStream file = new IFSTextFileOutputStream(as400, "/File");

                  // Write a String to the file.
                  // Because no CCSID was specified
                  // before writing to the file,
                  // Unicode characters will be
                  // written to the file. The file
                  // will be tagged as having Unicode
                  // data.
file.write("Hello world");

                  // Close the file.
file.close();
```

相關參考

『IFSFileWriter』
使用 IFSFileWriter 寫入整合檔案系統中的字元檔案。IFSFileWriter 的用途在於寫入字元的串流。

IFSFileWriter:

使用 IFSFileWriter 寫入整合檔案系統中的字元檔案。IFSFileWriter 的用途在於寫入字元的串流。

IFSFileWriter 會取代 IFSTextFileOutputStream。

範例：使用 IFSFileWriter

下列範例將說明 IFSFileWriter 的使用方式：

```
import java.io.PrintWriter;
 import java.io.BufferedWriter;
  // Work with /File1 on the system mysystem.
 AS400 as400 = new AS400("mysystem");
 IFSFile file = new IFSFile(system, "/File1");
 PrintWriter writer = new PrintWriter(new BufferedWriter(new IFSFileWriter(file)));
```

```
| // Write a line of text to the file, converting characters.
| writer.println(text);
| // Close the file.
| writer.close();
```

| **相關資訊**

| IFSFileWriter Javadoc

IFSRandomAccessFile:

IFSRandomAccessFile 類別代表伺服器上用來讀取及寫入資料的檔案。

IFSRandomAccessFile

Java 程式可循序或隨機地讀取及寫入資料。如同在 IFSFile 中，IFSRandomAccessFile 中的方法從 java.io 套件中複製 RandomAccessFile 的方法。除了這些方法以外，IFSRandomAccessFile 還有其他 iSeries 伺服器專用的方法。透過 IFSRandomAccessFile，Java 程式可以執行下列動作：

- 開啓檔案供讀取、寫入或讀取/寫入存取。Java 程式可選擇性地指定檔案共用模式及存在選項。
- 從現行位移讀取檔案中的資料。
- 從現行位移將資料寫入檔案。
- 取得或設定檔案的現行位移。
- 關閉檔案。

除了 java.io RandomAccessFile 中的方法以外，IFSRandomAccessFile 還可讓 Java 程式執行下列作業：

- 確定已寫入的磁碟位元組。
- 鎖定或解除鎖定檔案中的位元組。
- 鎖定和解除鎖定串流的位元組。詳細資訊，請參閱 IFSKey。
- 開啓檔案之後指定共用模式。詳細資訊，請參閱共用模式。
- 開啓檔案之後指定存在選項。Java 程式可選擇下列其中一項：
  - 如果檔案存在，則開啓此檔案；如果檔案不存在，則建立此檔案。
  - 如果檔案存在，則置換此檔案；如果檔案不存在，則建立此檔案。
  - 如果檔案存在，則無法開啓；如果檔案不存在，則建立此檔案。
  - 如果檔案存在，則開啓此檔案；如果檔案不存在，則無法開啓。
  - 如果檔案存在，則置換此檔案；如果檔案不存在，則無法開啓。

範例：使用 IFSRandomAccessFile

下面範例將告訴您如何使用 IFSRandomAccessFile 類別，以 1K 四個位元組寫入到檔案中。

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Open a file object that represents
                    // the file.
IFSRandomAccessFile aFile = new IFSRandomAccessFile(sys,"/mydir1/myfile", "rw");

                    // Establish the data to write.
byte i = 123;

                    // Write to the file 10 times at 1K
                    // intervals.
for (int j=0; j<10; j++)
{
```

```
                    // Move the current offset.
   aFile.seek(j * 1024);

                    // Write to the file. The current
                    // offset advances by the size of
                    // the write.
   aFile.write(i);
}

                    // Close the file.
aFile.close();
```

IFSFileDialog:

IFSFileDialog 類別可讓您遍訪檔案系統並選取檔案。

IFSFileDialog

此類別可使用 IFSFile 類別遍訪 iSeries 伺服器上的整合檔案系統中的目錄及檔案清單。此類別上的方法可讓 Java 程式設定關於對話按鈕的文字及設定過濾程式。請注意，也可以使用依據 Swing 1.1 的 IFSFileDialog 類別。

您可以透過 FileFilter 類別設定過濾程式。如果使用者在對話框中選取檔案，可以使用 getFileName() 方法來取得已選取的檔案名稱。可以使用 getAbsolutePath() 方法以取得已選取的檔案路徑與名稱。

範例：使用 IFSFileDialog

下面範例說明如何設定具有兩個過濾程式的對話，以及設定關於對話按鈕的文字。

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a dialog object setting
                    // the text of the dialog's title
                    // bar and the server to traverse.
IFSFileDialog dialog = new IFSFileDialog(this, "Title Bar Text", sys);

                    // Create a list of filters then set
                    // the filters in the dialog. The
                    // first filter will be used when
                    // the dialog is first displayed.
FileFilter[] filterList = {new FileFilter("All files (*.*)", "*.*"),
                           new FileFilter("HTML files (*.HTML", "*.HTM")};

dialog.setFileFilter(filterList, 0);

                    // Set the text on the buttons of
                    // the dialog.
dialog.setOkButtonText("Open");
dialog.setCancelButtonText("Cancel");

                    // Show the dialog. If the user
                    // selected a file by pressing the
                    // Open button, get the file the
                    // user selected and display it.
if (dialog.showDialog() == IFSFileDialog.OK)
   System.out.println(dialog.getAbsolutePath());
```

IFSKey 類別:

如果 Java 程式可讓其他程式同時存取檔案， 則 Java 程式可以在某期間內鎖定此檔案中的位元組。在該期間，此程式專用此檔案的該區段。

當鎖定順利完成時，整合檔案系統類別會傳回 IFSKey 物件。提供此物件給 unlock() 方法來指出要解除鎖定的位元組。關閉檔案之後，系統會解除鎖定仍在檔案中的全部鎖定 (系統會對程式沒有解除鎖定的每一個鎖定執行解除鎖定)。

範例：使用 IFSKey

下面範例說明如何使用 IFSKey 類別。

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Open an input stream. This
                    // constructor opens with share_all
                    // so other programs can open this
                    // file.
IFSFileInputStream aFile =
              new IFSFileInputStream(sys,"/mydir1/mydir2/myfile");

                    // Lock the first 1K bytes in the
                    // file. Now no other instance can
                    // read these bytes.
IFSKey key = aFile.lock(1024);

                    // Read the first 1K of the file.
byte data[] = new byte[1024];
aFile.read(data);

                    // Unlock the bytes of the file.
aFile.unlock(key);

                    // Close the file.
aFile.close();
```

檔案共用模式:

Java 程式可指定檔案開啓時的共用模式。此程式可讓其它程式同時開啓此檔案，或者讓此程式對此檔案具有專用存取權。

下面範例說明如何指定檔案共用模式。

```
                     // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Open a file object that
                    // represents the file. Since this
                    // program specifies share-none, all
                    // other open attempts fail until
                    // this instance is closed.
IFSFileOutputStream aFile = new IFSFileOutputStream(sys,
                                                    "/mydir1/mydir2/myfile",
                                                    IFSFileOutputStream.SHARE_NONE,
                                                    false);

                    // Perform operations on the file.

                    // Close the file. Now other open
                    // requests succeed.
aFile.close();
```

IFSSystemView:

IFSSystemView 提供 iSeries 整合檔案系統的閘道，用於建構 javax.swing.JFileChooser 物件時。

JFileChooser 是建置導覽及選擇檔案之對話框的標準 Java 方式。

範例：使用 IFSSystemView

下列範例將示範 IFSSystemView 的使用方式：

```
import com.ibm.as400.access.AS400;
import com.ibm.as400.access.IFSJavaFile;
import com.ibm.as400.access.IFSSystemView;
import javax.swing.JFileChooser;
import java.awt.Frame;

// Work with directory /Dir on the system myAS400.
AS400 system = new AS400("myAS400");
IFSJavaFile dir = new IFSJavaFile(system, "/Dir");
JFileChooser chooser = new JFileChooser(dir, new IFSSystemView(system));
Frame parent = new Frame();
int returnVal = chooser.showOpenDialog(parent);
if (returnVal == JFileChooser.APPROVE_OPTION) {
IFSJavaFile chosenFile = (IFSJavaFile)(chooser.getSelectedFile());
  System.out.println("You selected the file named " +
                     chosenFile.getName());
 }
```

相關資訊

IFSSystemView Javadoc

ISeriesNetServer

ISeriesNetServer 類別代表伺服器中的 NetServer 服務。此類別可讓使用者查詢及修改 NetServer 的狀態及配置。

ISeriesNetServer 置換 NetServer 類別。

相關資訊

ISeriesNetServer Javadoc

JavaApplicationCall

JavaApplicationCall 類別讓您有能力從您的用戶端，使用伺服器 JVM 來執行位於伺服器的 Java 程式。

JavaApplicationCall

在建立自用戶端到伺服器的連線後，JavaApplicationCall 類別可讓您配置下列：

1. 使用 setClassPath() 方法，在伺服器中設定 CLASSPATH 環境變數
2. 使用 setParameters() 方法，定義您程式的參數
3. 使用 run() 執行程式
4. 自用戶端傳送輸入至 Java 程式。Java 程式會透過以 sendStandardInString() 方法設定的標準輸入來讀取輸入。您可以透過 getStandardOutString() 及 getStandardErrorString()，將標準輸出及標準錯誤從 Java 程式重新導向至用戶端。

JavaApplicationCall 是一個可從 Java 程式呼叫的類別。然而，IBM Toolbox for Java 也提供了公用程式以呼叫位於伺服器的 Java 程式。這些公用程式是完整的 Java 程式，您可以從工作站加以執行。請參閱 RunJavaApplication 類別，以取得其餘相關資訊。

範例

JavaApplicationCall Javadoc 參考文件中的範例說明，如何從用戶端執行位於伺服器的程式 (此程式會輸出 "Hello World!")：

JavaApplicationCall

JDBC 類別

JDBC 是 Java 平台所附的應用程式設計介面 (API)，可讓 Java 程式連接多種資料庫。

如需 JDBC 及 IBM Toolbox for Java 支援 JDBC 的相關資訊 (包括目前進行的改良、JDBC 內容及未支援的 SQL 類型)，請參閱下列頁面：

第 290 頁的『JDBC』

支援的介面

下表列出所支援的 JDBC 介面，以及使用該介面必要的 API：

支援的 JDBC 介面	必要的 API
Blob 提供對二進位大型物件 (BOLB) 的存取	JDBC 2.1 基核
CallableStatement 執行 SQL 儲存程序	JDK 1.1
Clob 提供對字元大型物件 (CLOB) 的存取	JDBC 2.1 基核
Connection 代表對特定資料庫的連線	JDK 1.1
ConnectionPool 代表 Connection 物件儲存區。	JDBC 2.0 選用性套件
ConnectionPoolDataSource 代表儲存 AS400JDBCPooledConnection 物件的 Factory。	JDBC 2.0 選用性套件
DatabaseMetaData 提供有關資料庫的整體資訊。	JDK 1.1
DataSource 代表資料庫連線的 Factory。	JDBC 2.0 選用性套件
Driver 建立連線並傳回關於驅動程式版本的資訊。	JDK 1.1
ParameterMetaData 可讓您取得 PreparedStatement 物件中的參數之類型和內容相關資訊	JDBC 3.0 API
PreparedStatement 執行已編譯的 SQL 陳述式	JDK 1.1
ResultSet 提供對資料表的存取，這些資料是藉由執行 SQL 查詢或 DatabaseMetaData 編目方法所產生的	JDK 1.1
ResultSetMetaData 提供有關特定的 ResultSet 之資訊	JDK 1.1
RowSet 是一個封裝 ResultSet 的已連接列集	JDBC 2.0 選用性套件
Savepoint 可在異動內提供更細微的控制	JDBC 3.0 API
Statement 執行 SQL 陳述式並得到結果	JDK 1.1
XAConnection 是指在廣域 XA 異動中使用的資料庫連線	JDBC 2.0 選用性套件
XAResource 是在 XA 異動中使用的資源管理程式	JDBC 2.0 選用性套件

範例

下列範例說明使用 IBM Toolbox for Java JDBC 驅動程式的方法。

- 第 451 頁的『範例：使用 JDBCPopulate 以建立及輸入表格資料』
- 第 453 頁的『範例：使用 JDBCQuery 以查詢表格』

AS400JDBCBlob 類別:

您可使用 AS400JDBCBlob 物件存取二進位大型物件 (BLOB)，例如聲音位元組檔 (.wav) 或影像檔 (.gif)。

AS400JDBCBlob

AS400JDBCBlob 類別與 AS400JDBCBlobLocator 類別的最大不同之處在於 blob 的儲存處。透過 AS400JDBCBlob 類別，blob 會儲存在資料庫中，進而影響資料庫檔案的大小。AS400JDBCBlobLocator 類別會在資料庫檔案中儲存一個定位碼 (可視爲一個指標)，指向 blob 所在之處。

透過 AS400JDBCBlob 類別，可使用 lob 臨界特性。這個特性指定可擷取作爲部分結果集的最大大型物件 (LOB) 大小 (千位元組)。 當 LOB 大於這個臨界時，將使用伺服器的額外通訊，以片斷方式擷取它們。較大的 LOB 臨界雖可減少伺服器通訊的頻率，但它們會下載更多的 LOB 資料，即使用不到它們。使用較小的工作臨界，與伺服器的通訊就可能會更頻繁，但此時只會在必要的時候下載 LOB 資料。有關其他可用內容的資訊， 請參閱 JDBC 內容。

您可使用 AS400JDBCBlob 類別來執行下列事項：

- 傳回完整的二進位大型物件以作爲未解譯位元組的串流
- 傳回二進位大型物件部分的內容
- 傳回二進位大型物件的長度
- 建立二進位串流來寫入二進位大型物件
- 將位元組陣列寫入二進位大型物件
- 將全部或部分的位元組陣列寫入二進位大型物件
- 截斷二進位大型物件

範例

下列範例說明如何使用 AS400JDBCBlob 類別來讀取及更新二進位大型物件：

範例：使用 AS400JDBCBlob 類別來讀取二進位大型物件

```
Blob blob = resultSet.getBlob (1);
long length = blob.length ();
     byte[] bytes = blob.getBytes(1, (int) length);
```

範例：使用 AS400JDBCBlob 類別來更新二進位大型物件

```
ResultSet rs = statement.executeQuery ("SELECT BLOB FROM MYTABLE");
     rs.absolute(5);
     Blob blob = rs.getBlob(1);
               // Change the bytes in the blob, starting at the seventh byte
               // of the blob
     blob.setBytes (7, new byte[] { (byte) 57, (byte) 58, (byte) 98});
               //Update the blob in the result set, changing the blob starting
               // at the seventh byte of the blob (1-based) and truncating the
               // blob at the end of the updated bytes (the blob now has 9 bytes).
     rs.updateBlob(1, blob);
               // Update the database with the change.  This will change the blob
     // in the database starting at the seventh byte of the blob, and
               // truncating at the end of the updated bytes.
         rs.updateRow ();
     rs.close();
```

AS400JDBCBlobLocator 類別

您可使用 AS400JDBCBlobLocator 物件來存取二進位大型物件。

您可使用 AS400JDBCBlobLocator 類別來執行下列作業：

- 傳回完整的二進位大型物件以作爲未解譯位元組的串流
- 傳回二進位大型物件部分的內容
- 傳回二進位大型物件的長度

- 建立二進位串流來寫入二進位大型物件
- 將位元組陣列寫入二進位大型物件
- 將全部或部分的位元組陣列寫入二進位大型物件
- 截斷二進位大型物件

CallableStatement 介面:

您可以使用 CallableStatement 物件來執行 SQL 儲存程序。呼叫的儲存程序必須已儲存在資料庫。CallableStatement 沒有儲存程序，它僅會呼叫儲存程序。

CallableStatement

儲存程序可傳回一或多個 ResultSet 物件，而且可使用 IN 參數、OUT 參數以及 INOUT 參數。使用 Connection.prepareCall() 來建立新的 CallableStatement 物件。

CallableStatement 物件可讓您以單一群組的方式，透過批次支援的使用，對資料庫提出多重 SQL 指令。您可以使用批次支援來獲得較佳的效能，因爲處理一組作業通常比一次處理一個作業更快。關於使用批次支援的詳細資訊，請參閱 JDBC 支援的加強功能。

CallableStatement 可讓您依名稱取得及設定參數和直欄，然而使用直欄索引的效能較佳。

範例：使用 CallableStatement

下面範例說明如何使用 CallableStatement 介面。

```
                    // Connect to the server.
Connection c = DriverManager.getConnection("jdbc:as400://mySystem");

                    // Create the CallableStatement
                    // object. It precompiles the
                    // specified call to a stored
                    // procedure. The question marks
                    // indicate where input parameters
                    // must be set and where output
                    // parameters can be retrieved.
                    // The first two parameters are
                    // input parameters, and the third
                    // parameter is an output parameter.
CallableStatement cs = c.prepareCall("CALL MYLIBRARY.ADD (?, ?, ?)");

                    // Set input parameters.
cs.setInt (1, 123);
cs.setInt (2, 234);

                    // Register the type of the output
                    // parameter.
cs.registerOutParameter (3, Types.INTEGER);

                    // Run the stored procedure.
cs.execute ();

                    // Get the value of the output
                    // parameter.
int sum = cs.getInt (3);

                    // Close the CallableStatement and
                    // the Connection.
cs.close();
c.close();
```

AS400JDBCClob 類別:

您可以使用 AS400JDBCClob 物件來存取字元大型物件 (CLOB)，例如大型文件。

AS400JDBCClob

AS400JDBCClob 類別與 AS400JDBCClobLocator 類別的最大不同之處在於 blob 的儲存處。透過 AS400JDBCClob 類別，clob 是儲存在資料庫中，進而擴增資料庫檔案的大小。AS400JDBCClobLocator 類別會在資料庫檔案中儲存一個定位碼 (可視爲一個指標)，指向 clob 所在之處。

透過 AS400JDBCClob 類別，可使用 lob 臨界特性。這個特性指定可擷取作爲部分結果集的最大大型物件 (LOB) 大小 (千位元組)。 當 LOB 大於這個臨界時，將使用伺服器的額外通訊，以片斷方式擷取它們。較大的 LOB 臨界雖可減少伺服器通訊的頻率，但它們會下載更多的 LOB 資料，即使用不到它們。 使用較小的 lob 臨界，與伺服器的通訊就可能會更頻繁，但此時只會在必要的時候下載 LOB 資料。有關其他可用內容的資訊， 請參閱第 295 頁的『IBM Toolbox for Java JDBC 內容』。

使用 AS400JDBCClob 類別，您可進行下列作業：

- 傳回完整的 clob 作爲 ASCII 字元串流
- 傳回 clob 的內容作爲字元串流
- 傳回 clob 的部分內容
- 傳回 clob 的長度
- 建立 Unicode 字元串流或 ASCII 字元串流以寫入 clob
- 將字串寫入 clob
- 截斷 clob

範例

下列範例說明如何使用 AS400JDBCClob 類別來讀取 clob 及更新 clob：

範例：使用 AS400JDBCClob 類別來讀取 clob

```
Clob clob = rs.getClob (1);
     int length = clob.length();
     String s = clob.getSubString(1, (int) length);
```

範例：使用 AS400JDBCClob 類別來更新 clob

```
ResultSet rs = statement.executeQuery ("SELECT CLOB FROM MYTABLE");
     rs.absolute(4);
Clob clob = rs.getClob (1);

               // Change the characters in the clob, starting at the third character
               // of the clob
clob.setString (3, "Small");

               // Update the clob in the result set, starting at the third character
               // of the clob and truncating the clob at the end of the update string
               // (the clob now has 7 characters).
     rs.updateClob(1, clob);

               // Update the database with the updated clob.  This will change the
     // clob in the database starting at the third character of the clob,
               // and truncating at the end of the update string.
         rs.updateRow ();
     rs.close();
```

AS400JDBCClobLocator 類別

您可使用 AS400JDBCClobLocator 物件來存取字元大型物件 (CLOB)。

使用 AS400JDBCClobLocator 類別，您可進行下列作業：

- 傳回完整的 clob 作爲 ASCII 字元串流
- 傳回完整的 clob 作爲字元串流
- 傳回 clob 的部分內容
- 傳回 clob 的長度
- 建立 Unicode 字元串流或 ASCII 字元串流以寫入 clob
- 將字串寫入 clob
- 截斷 clob

AS400JDBCConnection 類別:

AS400JDBCConnection 類別可提供特定 DB2 UDB for iSeries 資料庫的 JDBC 連線。

使用 DriverManager.getConnection() 來建立新的 AS400JDBCConnection 物件。詳細資訊，請參閱第 66 頁的『登記 JDBC 驅動程式』。

在建立連線時可以指定許多可選用的內容。可將內容指定爲 URL 的一部分，或是在 java.util.Properties 物件中指定。如需 AS400JDBCDriver 支援的內容完整清單，請參閱第 295 頁的『IBM Toolbox for Java JDBC 內容』。

註: 一個連線最多可包含 9999 個 open 陳述式。

AS400JDBCConnection 包括儲存點及陳述式層次保留功能的支援，以及傳回自動產生索引的限制支援。有關這些和其他加強功能的資訊，請參閱第 292 頁的『版本 5 版次 3 的 JDBC 支援加強功能』。

若要使用 Kerberos 通行證，僅須在 JDBC URL 物件上設定系統名稱 (而非密碼)。 使用者身分是透過 Java 一般安全性服務 (JGSS) 組織架構擷取的，因此您也不必在 JDBC URL 中指定使用者。在 AS400JDBCConnection 物件中一次只能設定一種鑑別方法。設定密碼會清除 Kerberos 通行證或設定檔記號。詳細資訊，請參閱第 23 頁的『AS400 類別』和 J2SDK, v1.4 Security Documentation  。

使用 AS400JDBCConnection 類別，您可以執行下列作業：

- 建立陳述式 (Statement、PreparedStatement 或 CallableStatement 物件)
- 建立陳述式，該陳述式具有特定的結果設定類型及並行 (Statement、PreparedStatement 或 CallableStatement 物件)
- 對資料庫的確定及回轉變更，以及釋放目前已保留的資料庫鎖定
- 關閉連線，立即關閉伺服器資源，而不是等到它們自動釋放
- 針對連線的設定保留功能及取得保留功能
- 連線的設定異動隔離及取得異動隔離
- 連線的取得 meta 資料
- 開啓或關閉設定自動確定
- 取得主電腦伺服器工作的工作 ID，該工作對應於連線

如果使用 JDBC 3.0 並連接到正在執行 i5/OS V5R2 或更新版本的伺服器，您可以使用 AS400JDBCConnection 執行下列動作：

- 建立具有特定結果設定保留功能的陳述式 (Statement、PreparedStatement 或 CallableStatement 物件)
- 建立會傳回自動產生的索引之預備陳述式 (在 Statement 物件上呼叫 getGeneratedKeys() 時)
- 使用儲存點，它透過異動提供更多的細微性控制：
  - 設定儲存點
  - 回轉儲存點
  - 釋放儲存點

AS400JDBCConnectionPool:

AS400JDBCConnectionPool 類別代表 AS400JDBCConnection 物件的儲存區，Java 程式可以使用這些物件，作爲 IBM Toolbox for Java 支援 JDBC 2.0 Optional Package API 的一部分。

您可使用 AS400JDBCConnectionPoolDataSource 來指定在儲存區中所建立之連線的內容，如下列範例所示。

在要求連線之後而且儲存區正在使用的情形下，您無法變更連線儲存區資料來源。若要重設連線儲存區資料來源，首先呼叫儲存區上的 close()。

使用 AS400JDBCConnection 物件上的 close()，將連線傳回 AS400JDBCConnectionPool。

註: 如果沒有將連線傳回儲存區，則連線儲存區的大小會繼續成長且無法重新使用連線。

使用由 ConnectionPool 所繼承的方法來設定儲存區上的內容。您可設定的部分內容如下：
- 儲存區中可允許的最大連線數
- 連線的最大生命週期
- 連線的最大休止時間

您還可使用 Java Naming and Directory Interface(TM) (JNDI) 服務提供者來登記 AS400JDBCConnectionPoolDataSource 物件。如需 JNDI 服務提供者的相關資訊，請參閱 IBM Toolbox for Java 參照鏈結。

範例：使用連線儲存區

下列範例從 JNDI 取得連線儲存區資料來源，並用它來建立一個有 10 個連線的連線儲存區：

```
                // Obtain an AS400JDBCConnectionPoolDataSource object from JNDI
                // (assumes JNDI environment is set).
        Context context = new InitialContext(environment);
        AS400JDBCConnectionPoolDataSource datasource =
        (AS400JDBCConnectionPoolDataSource)context.lookup("jdbc/myDatabase");

                // Create an AS400JDBCConnectionPool object.
        AS400JDBCConnectionPool pool = new AS400JDBCConnectionPool(datasource);

                // Adds 10 connections to the pool that can be used by the
                // application (creates the physical database connections based on
                // the data source).
        pool.fill(10);

                // Get a handle to a database connection from the pool.
        Connection connection = pool.getConnection();

    ... Perform miscellaneous queries/updates on the database.

                // Close the connection handle to return it to the pool.
        connection.close();
```

```
... Application works with some more connections from the pool.

    // Close the pool to release all resources.
    pool.close();
```

DatabaseMetaData 介面:

您可以使用 DatabaseMetaData 物件取得關於整個資料庫的資訊以及型錄資訊。

下面範例說明如何傳回表格清單，它是一個分目功能：

```
                  // Connect to the server.
Connection c = DriverManager.getConnection("jdbc:as400://mySystem");

                  // Get the database metadata from the connection.
DatabaseMetaData dbMeta = c.getMetaData();

                  // Get a list of tables matching the following criteria.
String catalog = "myCatalog";
String schema  = "mySchema";
String table   = "myTable%"; // % indicates search pattern
String types[]  = {"TABLE", "VIEW", "SYSTEM TABLE"};
ResultSet rs = dbMeta.getTables(catalog, schema, table, types);

                  // Iterate through the ResultSet to get the values.

                  // Close the Connection.
c.close();
```

AS400JDBCDataSource 類別:

AS400JDBCDataSource 類別代表 iSeries 資料庫連線的 Factory。AS400JDBCConnectionPoolDataSource 類別代表 AS400JDBCPooledConnection 物件的 Factory。

您可以使用 Java Naming and Directory Interface (JNDI) 服務提供者，登記任一種類的資料來源物件。如需 JNDI 服務提供者的相關資訊，請參閱 IBM Toolbox for Java 參照鏈結。

範例

下列範例說明建立及使用 AS400JDBCDataSource 物件的方法。最後的兩個範例顯示如何以 JNDI 登記一個 AS400JDBCDataSource 物件，然後使用 JNDI 傳回的物件取得資料庫連線。請注意，即使使用不同的 JNDI 服務提供者，其程式碼都十分類似。

範例：建立 AS400JDBCDataSource 物件

下列範例顯示如何建立 AS400JDBCDataSource 物件並將之與資料庫連接：

```
// Create a data source for making the connection.
AS400JDBCDataSource datasource = new AS400JDBCDataSource("myAS400");
datasource.setUser("myUser");
datasource.setPassword("MYPWD");

// Create a database connection to the iSeries.
Connection connection = datasource.getConnection();
```

範例：建立可用來快取 JDBC 連線的 AS400JDBCConnectionPoolDataSource 物件

下列範例將告訴您如何使用 AS400JDBCConnectionPoolDataSource 快取 JDBC 連線。

```
        // Create a data source for making the connection.
AS400JDBCConnectionPoolDataSource dataSource = new AS400JDBCConnectionPoolDataSource("myAS400");
datasource.setUser("myUser");
```

```
datasource.setPassword("MYPWD");

// Get the PooledConnection.
PooledConnection pooledConnection = datasource.getPooledConnection();
```

範例：使用 JNDI 服務提供者類別來儲存 AS400JDBCDataSource 物件

下面範例說明如何使用 JNDI 服務提供者類別，將 DataSource 物件直接儲存到伺服器上的整合檔案系統：

```
        // Create a data source to the iSeries database.
        AS400JDBCDataSource dataSource = new AS400JDBCDataSource();
        dataSource.setServerName("myAS400");
dataSource.setDatabaseName("myAS400 Database");

        // Register the datasource with the Java Naming and Directory Interface (JNDI).
        Hashtable env = new Hashtable();
        env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.fscontext.RefFSContextFactory");
        Context context = new InitialContext(env);
        context.bind("jdbc/customer", dataSource);

        // Return an AS400JDBCDataSource object from JNDI and get a connection.
        AS400JDBCDataSource datasource = (AS400JDBCDataSource) context.lookup("jdbc/customer");
        Connection connection = datasource.getConnection("myUser", "MYPWD");
```

範例：以 AS400JDBCDataSource 物件及 IBM SecureWay Directory 類別，搭配使用 Lightweight Directory Access Protocol (LDAP) 目錄伺服器

下列範例說明如何使用 IBM SecureWay® Directory 類別將物件儲存到 Lightweight Directory Access Protocol (LDAP) 目錄伺服器中：

```
        // Create a data source to the iSeries database.
        AS400JDBCDataSource dataSource = new AS400JDBCDataSource();
        dataSource.setServerName("myAS400");
dataSource.setDatabaseName("myAS400 Database");

// Register the datasource with the Java Naming and Directory Interface (JNDI).
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY, "com.ibm.jndi.LDAPCtxFactory");
Context context = new InitialContext(env);
context.bind("cn=myDatasource, cn=myUsers, ou=myLocation,o=myCompany,c=myCountryRegion",
             dataSource);

// Return an AS400JDBCDataSource object from JNDI and get a connection.
AS400JDBCDataSource datasource = (AS400JDBCDataSource) context.lookup(
   "cn=myDatasource, cn=myUsers, ou=myLocation,o=myCompany,c=myCountryRegion");
Connection connection = datasource.getConnection("myUser", "MYPWD");
```

登記 JDBC 驅動程式:

使用 JDBC 存取伺服器資料庫檔案中的資料之前，必須以 DriverManager 登記 IBM Toolbox for Java 授權程式的 JDBC 驅動程式。您可以使用 Java 系統內容來登記驅動程式，或由 Java 程式登記該驅動程式：

- 使用系統特性來登記

  每一個虛擬機器都有自己設定系統特性的方法。例如，來自 JDK 的 Java 指令使用 -D 選項來設定系統內容。若要使用系統特性設定驅動程式，請指定下列：

  ```
  "-Djdbc.drivers=com.ibm.as400.access.AS400JDBCDriver"
  ```

- 使用 Java 程式進行登記

  若要載入 IBM Toolbox for Java JDBC 驅動程式，請在第一次 JDBC 呼叫之前，將下列程式碼新增至 Java 程式：

```
Class.forName("com.ibm.as400.access.AS400JDBCDriver");
```

IBM Toolbox for Java JDBC 驅動程式在載入時會自行登記，這是登記驅動程式的優先方法。您也可以使用下列程式碼來明確登記 IBM Toolbox JDBC 驅動程式：

```
java.sql.DriverManager.registerDriver (new com.ibm.as400.access.AS400JDBCDriver ());
```

IBM Toolbox for Java JDBC 驅動程式不像從伺服器取得資料的其他 IBM Toolbox for Java 類別，它不需使用 AS400 物件作爲輸入參數。不過，內部使用 AS400 物件來管理預設使用者和密碼快取。第一次連接伺服器時，可能會提示使用者輸入使用者 ID 與密碼。使用者可選擇儲存此使用者 ID 作爲預設使用者 ID，然後新增密碼到密碼快速記憶體。與其他 IBM Toolbox for Java 功能一樣，如果由 Java 程式提供使用者 ID 及密碼，則不設定預設使用者，也不快取密碼。有關管理連線的資訊，請參閱第 399 頁的『管理連線』。

使用 JDBC 驅動程式連接伺服器上的資料庫

您可以使用 DriverManager.getConnection() 方法來連接伺服器資料庫。DriverManager.getConnection() 採用一致資源定位器 (URL) 字串作爲引數。JDBC 驅動程式管理程式會試圖尋找一驅動程式，因爲該驅動程式可連接至由此 URL 代表的資料庫。使用 IBM Toolbox for Java 驅動程式時，請使用下列 URL 的語法：

```
"jdbc:as400://systemName/defaultSchema;listOfProperties"
```

註: URL 可以省略 systemName 或 defaultSchema。

若要使用 Kerberos 通行證，僅須在 JDBC URL 物件上設定系統名稱 (而非密碼)。 使用者身分是透過 Java 一般安全性服務 (JGSS) 組織架構擷取的，因此您也不必在 JDBC URL 中指定使用者。在 AS400JDBCConnection 物件中一次只能設定一種鑑別方法。設定密碼會清除 Kerberos 通行證或設定檔記號。詳細資訊，請參閱第 23 頁的『AS400 類別』和 J2SDK, v1.4 Security Documentation  。

範例：使用 JDBC 驅動程式來連接伺服器

範例：使用未指定系統名稱的 URL

此範例將說明如何提示使用者鍵入想要連接的系統名稱。

```
                    // Connect to unnamed system.
                    // User receives prompt to type system name.
Connection c = DriverManager.getConnection("jdbc:as400:");
```

範例：連接到伺服器資料庫；未指定預設網目或內容

```
                    // Connect to system 'mySystem'. No
                    // default schema or properties are
                    // specified.
Connection c  = DriverManager.getConnection("jdbc:as400://mySystem");
```

範例：連接到伺服器資料庫；已指定預設網目

```
                    // Connect to system 'mySys2'. The
                    // default schema 'myschema' is
                    // specified.
Connection c2 = DriverManager.getConnection("jdbc:as400://mySys2/mySchema");
```

範例：連接到伺服器資料庫並使用 java.util.Properties 來指定內容

Java 程式可指定一組 JDBC 內容，其方法是使用 java.util.Properties 介面或指定內容作爲 URL 的一部分。如需支援的內容清單，請參閱 第 295 頁的『IBM Toolbox for Java JDBC 內容』。

例如，若要使用 Properties 介面指定特性，請使用下列程式作爲範例：

```
                    // Create a properties object.
Properties p = new Properties();

                    // Set the properties for the
                    // connection.
p.put("naming", "sql");
p.put("errors", "full");

                    // Connect using the properties
                    // object.
Connection c = DriverManager.getConnection("jdbc:as400://mySystem",p);
```

範例：連接到伺服器資料庫並使用「全球資源定位器 (URL)」來指定內容

```
                    // Connect using properties. The
                    // properties are set on the URL
                    // instead of through a properties
                    // object.
Connection c = DriverManager.getConnection(
                     "jdbc:as400://mySystem;naming=sql;errors=full");
```

範例：連接到伺服器資料庫並指定使用者 ID 和密碼

```
                    // Connect using properties on the
                    // URL and specifying a user ID and
                    // password
Connection c = DriverManager.getConnection(
                     "jdbc:as400://mySystem;naming=sql;errors=full",
                     "auser",
                     "apassword");
```

範例：切斷與資料庫的連線若要切斷與伺服器的連線，請對 Connecting 物件使用 close() 方法。使用下列陳述式來關閉在前一個範例中建立的連線：

```
c.close();
```

AS400JDBCParameterMetaData 類別:

AS400JDBCParameterMetaData 類別可讓您的程式擷取 PreparedStatement

AS400JDBCParameterMetaData 提供的方法，可讓您執行下列作業：

- 取得參數的類別名稱
- 取得 PreparedStatement 中的參數數目
- 取得參數的 SQL 類型
- 取得參數的資料庫專屬類型名稱
- 取得參數的精準度或參數的小數位數

範例：使用 AS400JDBCParameterMetaData

下列範例告訴您一種使用 AS400JDBCParameterMetaData 從動態產生的 PreparedStatement 物件擷取參數之方法：

```
// Get a connection from the driver.
     Class.forName("com.ibm.as400.access.AS400JDBCDriver");
     Connection connection =
   DriverManager.getConnection("jdbc:as400://myAS400", "myUserId", "myPassword");

     // Create a prepared statement object.
     PreparedStatement ps =
   connection.prepareStatement("SELECT STUDENTS FROM STUDENTTABLE WHERE STUDENT_ID= ?");

     // Set a student ID into parameter 1.
     ps.setInt(1, 123456);
```

```
// Retrieve the parameter meta data for the prepared statement.
ParameterMetaData pMetaData = ps.getParameterMetaData();

// Retrieve the number of parameters in the prepared statement.
// Returns 1.
int parameterCount = pMetaData.getParameterCount();

// Find out what the parameter type name of parameter 1 is.
// Returns INTEGER.
String getParameterTypeName = pMetaData.getParameterTypeName(1);
```

PreparedStatement 介面:

當 SQL 陳述式將執行許多次時，您可以使用 PreparedStatement 物件。可前置編譯 SQL 陳述式。″prepared.″ 陳述式是已經過前置編譯的 SQL 陳述式。 此方法比使用某 Statement 物件執行同一個陳述式數次來得更有效率，因爲後者在每次執行此陳述式時都要編譯陳述式。此外，包含在 PreparedStatement 物件的 SQL 陳述式可能有一個或多個 IN 參數。使用 Connection.prepareStatement() 來建立 PreparedStatement 物件。

PreparedStatement 物件可讓您以單一群組的方式，透過批次支援的使用，對資料庫提出多重 SQL 指令。您可以使用批次支援來提升效能，因爲處理一組作業通常比一次處理一個作業還要快。關於使用批次支援的詳細資訊，請參閱 JDBC 支援的加強功能。

範例：使用 PreparedStatement

下面範例說明如何使用 PreparedStatement 介面。

```
                    // Connect to the server.
Connection c = DriverManager.getConnection("jdbc:as400://mySystem");

                    // Create the PreparedStatement
                    // object. It precompiles the
                    // specified SQL statement. The
                    // question marks indicate where
                    // parameters must be set before the
                    // statement is run.
PreparedStatement ps = c.prepareStatement("INSERT INTO MYLIBRARY.MYTABLE (NAME, ID) VALUES (?, ?)");

                    // Set parameters and run the
                    // statement.
ps.setString(1, "JOSH");
ps.setInt(2, 789);
ps.executeUpdate();

                    // Set parameters and run the
                    // statement again.
ps.setString(1, "DAVE");
ps.setInt(2, 456);
ps.executeUpdate();

                    // Close PreparedStatement and the
                    // Connection.
ps.close();
c.close();
```

ResultSet 類別:

您可以使用 ResultSet 物件來存取一個因執行查詢而產生的資料表格。會依順序擷取表格列。在一列中，可依任何次序來存取直欄值。

儲存在 ResultSet 中的資料利用各種不同的 get 方法加以擷取，視擷取的資料類型而定。next() 方法是用來移到下一列。

ResultSet 可讓您依名稱來取得及更新直欄，然而使用直欄索引結果可提升效能。

游標移動

游標是一種內部指標，可供結果集用來指向結果集內正由 Java 程式存取的橫列。

已提升 getRow() 方法的效能。在 V5R2 之前，使用含有負值的 ResultSet.last()、ResultSet.afterLast() 和 ResultSet.absolute() 會使現行列號變成無法使用。以前的限制已取消，因此 getRow() 方法能夠充分地發揮作用。

JDBC 2.0 與以上的版本的 JDBC 規格提供其它方法來存取資料庫內的特定位置：

可捲動的游標位置	
absolute	isFirst
afterLast	isLast
beforeFirst	last
first	moveToCurrentRow
getRow	moveToInsertRow
isAfterLast	previous
isBeforeFirst	relative

捲動功能

如果透過執行一個陳述式來建立結果集，您可以向後 (最後一個到第一個) 或向前 (第一個到最後一個) 移動 (捲動) 表格中的橫列。

支援這個移動的結果集稱爲可捲動的結果集。可捲動的結果集也支援絕對位置。絕對位置可讓您經由在結果集中指定一個位置，直接移到該橫列。

透過 JDBC 2.0 與以上的版本的 JDBC 規格，您有兩個額外捲動功能可在使用 ResultSet 類別時使用：不可捲動及可捲動結果集。

不可捲動結果集通常不會感應到當開啓它時所做的變更，而可捲動結果集則可感應到變更。

註: 對於可捲動的 insensitive 游標，IBM iSeries Server 只允許唯讀存取。如果結果集並行處理是唯讀的，則 IBM Toolbox for Java 支援可捲動的 insensitive 游標。如果結果集類型指定爲 insensitive 而且並行處理指定爲可更新，則結果集類型會變更爲 sensitive 並發出警告。

可更新的結果集

在應用程式中，您可以使用的結果集包括採用唯讀並行處理的結果集 (不能更新資料)，或採用可更新並行處理的結果集 (允許更新資料，並使用資料庫寫入鎖定來控制不同異動對相同資料項目的存取) 。在可更新的結果集中，可以更新、插入及刪除列。有不少更新方法可供您在程式中使用，例如：

- 更新 ASCII 串流
- 更新 Big Decimal
- 更新二進位串流

請參閱方法總結，以取得透過 ResultSet 介面可用的更新方法的完整報表。

範例：可更新的結果集

下面範例將告訴您如何使用當開啓時容許變更資料 (更新並行處理)，以及容許變更結果集 (可捲動) 的結果集。

```
                    // Connect to the server.
Connection c = DriverManager.getConnection("jdbc:as400://mySystem");

                    // Create a Statement object.  Set the result set
                    // concurrency to updatable.
Statement s = c.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);

                    // Run a query. The result is placed
                    // in a ResultSet object.
ResultSet rs = s.executeQuery ("SELECT NAME,ID FROM MYLIBRARY.MYTABLE FOR UPDATE");

                    // Iterate through the rows of the ResultSet.
                    // As we read the row, we will update it with
                    // a new ID.
int newId = 0;
while (rs.next ())
{

                    // Get the values from the ResultSet.
                    // The first value is a string, and
                    // the second value is an integer.
    String name = rs.getString("NAME");
    int id = rs.getInt("ID");

    System.out.println("Name = " + name);
    System.out.println("Old id = " + id);

                    // Update the id with a new integer.
    rs.updateInt("ID", ++newId);

                    // Send the updates to the server.
    rs.updateRow ();

    System.out.println("New id = " + newId);
}

                    // Close the Statement and the
                    // Connection.
s.close();
c.close();
```

ResultSetMetaData

ResultSetMetaData 介面決定 ResultSet 中的直欄類型和內容。

連線到執行 i5/OS V5R2 或更新版本的伺服器時，若使用延伸的 Meta 資料內容，可讓您增加下列 ResultSetMetaData 方法的正確性：

- getColumnLabel(int)
- isReadOnly(int)
- isSearchable(int)
- isWriteable(int)

此外，設定此內容爲 TRUE 會啓用 ResultSetMetaData.getSchemaName(int) 方法的支援。請注意，使用延伸的 meta 資料內容時必須從伺服器擷取更多資訊，因此可能會降低效能。

AS400JDBCRowSet 類別:

AS400JDBCRowSet 類別代表封裝一個 JDBC 結果集的已連接列集。AS400JDBCRowSet 的方法非常類似 AS400JDBCResultSet 的方法。使用時，資料庫會與之保持連線。

您可以使用 AS400JDBCDataSource 案例 或 AS400JDBCConnectionPoolDataSource 案例來建立連線，以連接到您要用來存取 AS400JDBCRowSet 資料的資料庫。

範例

下列範例告訴您如何使用 AS400JDBCRowSet 類別。

範例：建立、輸入和更新 AS400JDBCRowSet 物件

```
DriverManager.registerDriver(new AS400JDBCDriver());
// Establish connection by using a URL.
AS400JDBCRowSet rowset = new AS400JDBCRowSet("jdbc:as400://mySystem","myUser", "myPassword");

// Set the command used to populate the list.
rowset.setCommand("SELECT * FROM MYLIB.DATABASE");

// Populate the rowset.
rowset.execute();

// Update the customer balances.
while (rowset.next())
{
   double newBalance = rowset.getDouble("BALANCE") +
                       july_statements.getPurchases(rowset.getString("CUSTNUM"));
   rowset.updateDouble("BALANCE", newBalance);
   rowset.updateRow();
}
```

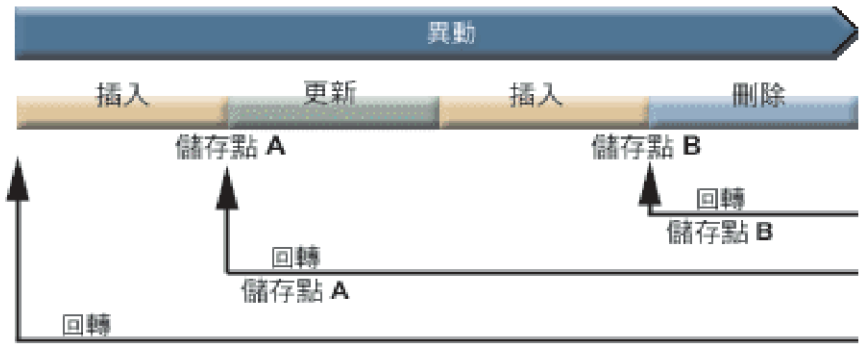
範例：從 JNDI 取得資料來源時，建立及植入 AS400JDBCRowSet 物件

```
// Get the data source that is registered in JNDI (assumes JNDI environment is set).
Context context = new InitialContext();
AS400JDBCDataSource dataSource = (AS400JDBCDataSource) context.lookup("jdbc/customer");

AS400JDBCRowSet rowset = new AS400JDBCRowSet();
// Establish connection by setting the data source name.
rowset.setDataSourceName("jdbc/customer");
rowset.setUsername("myuser");
rowset.setPassword("myPasswd");

// Set the prepared statement and initialize the parameters.
rowset.setCommand("SELECT * FROM MYLIBRARY.MYTABLE WHERE STATE = ? AND BALANCE > ?");
rowset.setString(1, "MINNESOTA");
rowset.setDouble(2, MAXIMUM_LIMIT);

// Populate the rowset.
rowset.execute();
```

AS400JDBCSavepoint 類別:

AS400JDBCSavepoint 類別代表異動中的邏輯岔斷點。使用 savepoints 可讓您對回轉異動時影響的變更進行更精確的控制。

圖 1：使用儲存點來控制異動中的回轉



例如，「圖 1」顯示一筆包含兩個儲存點 (A 及 B) 的異動。將異動回轉至其中一個儲存點僅會還原 (或反轉) 回轉呼叫點至儲存點的變更。這可防止還原整個異動中之所有的變更。請注意，一旦回轉至儲存點 A，以後就無法回轉至儲存點 B。在工作回轉超過儲存點 B 之後， 就無法存取儲存點 B。

範例：使用儲存點

在此實務範例中，假設您以應用程式更新學生的成績。在更新每筆學生成績之特定欄位的最後，您執行確定。您的程式碼偵測到與更新此欄位相關聯的特定錯誤，並在錯誤發生時回轉已進行的作業。而您知道此特定錯誤只會影響對目前成績進行的作業。

因此，您在每一次更新學生成績之間設定一個儲存點。現在，當此錯誤發生時，您僅須回轉學生表格中的前次更新。毋需回轉大量的作業，現在您只需回轉少量的作業。

以下程式碼範例說明如何使用儲存點。範例假設 John 的學生 ID 爲 123456， 而 Jane 的學生 ID 爲 987654。

```
// Get a connection from the driver
     Class.forName("com.ibm.as400.access.AS400JDBCDriver");

     // Get a statement object
 Statement statement = connection.createStatement();

     // Update John's record with his 'B' grade in gym.
int rows = statement.executeUpdate(
  "UPDATE STUDENTTABLE SET GRADE_SECOND_PERIOD = 'B' WHERE STUDENT_ID= '123456'");

     // Set a savepoint marking an intermediate point in the transaction
     Savepoint savepoint1 = connection.setSavepoint("SAVEPOINT_1");

     // Update Jane's record with her 'C' grade in biochemistry.
int rows = statement.executeUpdate(
   "UPDATE STUDENTTABLE SET GRADE_SECOND_PERIOD = 'C' WHERE STUDENT_ID= '987654'");

     // An error is detected, so we need to roll back Jane's record, but not John's.
     // Rollback the transaction to savepoint 1. The change to Jane's record is
     // removed while the change to John's record remains.
     connection.rollback(savepoint1);

     // Commit the transaction; only John's 'B' grade is committed to the database.
     connection.commit();
```

注意事項與限制

使用儲存點時必須注意到下列注意事項與限制：

注意事項

有關回轉如何影響游標及保留鎖定的資料庫規則，IBM Toolbox for Java 都會加以遵循。例如，將連線選項設定爲在一般回轉後保持游標開啓時，則在回轉至儲存點後游標仍會開啓。換句話說，當回轉要求牽涉到儲存點時，如果基礎資料庫不支援移動或關閉游標，IBM Toolbox for Java 就不會移動或關閉游標。

使用儲存點來回轉異動僅會還原從開始回轉點至儲存點執行的作業。在該儲存點之前執行的動作仍然存在。如同在先前的範例中一樣，請注意您可確定的異動包括回轉至特定的儲存點之前執行的作業，但不包括回轉至儲存點之後執行的作業。

異動確定時或整個異動已回轉時，會解除所有的儲存點，且這些儲存點也會變成無效。您也可以下列方法解除儲存點：呼叫 Connection.releaseSavepoint()。

限制

使用儲存點時有下列限制：

- 儲存點的名稱必須唯一。
- 在釋放、確定或回轉某儲存點之後，才能再次使用此儲存點的名稱。
- 自動確定必須設定爲「關閉」，儲存點才會有效。將自動確定設定爲「關閉」的方法爲：使用 `Connection.setAutoCommit(false)`。使用儲存點時啓用自動確定系統會發出異常訊息。
- 儲存點在整個 XA 連線過程中均無效。在 XA 連線時使用儲存點系統會發出異常訊息。
- 伺服器必須執行 i5/OS 版本 5 版次 2 或更新版本。當您連接 (或已連接) 到執行 V5R1 或之前版本 i5/OS 的伺服器時，若使用儲存點，就會丟出異常。

圖 1 長說明：使用儲存點來控制異動中的回轉 (rzahh586.gif):

位於 IBM Toolbox for Java：AS400JDBCSavepoint 類別

本圖說明如何使用儲存點來控制異動中的回轉。

說明

此圖由下列項目所構成：

- 藍色水平箭頭，指向右邊，標示著「異動」。「異動」箭頭代表從左邊開始到右邊結束的線形異動。
- 「異動」箭頭底下是和「異動」箭頭等長的多色列。此列分成四個顏色區段，從左到右代表組成異動的各個動作。列底下的兩個標籤代表異動中的儲存點。
- 多色列的下面有三個互相重疊的箭頭。箭頭指向左邊，每個箭頭代表將異動回轉至不同的點。

「異動」箭頭代表從左邊開始到右邊結束的異動。異動是由一系列的各個動作所組成 (多色列的不同部分)。從左到右的顏色區段代表如下：

- 第一個動作 (褐色區段) 標示「插入」
- 第二個動作 (綠色區段) 標示「更新」
- 第三個動作 (另一個褐色區段) 標示「插入」
- 第四個也是最後一個動作 (藍色區段) 標示「刪除」

多色列下面的標籤代表儲存點。第一個動作結束而第二個動作開始所在的點標示爲「儲存點 A」。第三個動作結束而第四個動作開始所在的點標示爲「儲存點 B」。

多色列下面的箭頭指向左邊，代表儲存點對回轉異動的影響程度：

- 第一個箭頭指向「儲存點 B」。將異動回轉到「儲存點 B」時，只會回轉最後一個動作 (藍色區段標示「刪除」)

- 第二個箭頭指向「儲存點 A」。將異動回轉到「儲存點 A」時，會從第四個動作回轉到第二個動作 (綠色區段標示「更新」，第二個褐色區段標示「插入」，藍色區段標示「刪除」)
- 最後一個箭頭指向異動的開頭。完全回轉異動時，會反轉所有的個別動作

以 Statement 物件執行 SQL 陳述式:

使用 Statement 物件執行 SQL 陳述式和選用性地取得它產生的 ResultSet。

PreparedStatement 繼承 Statement， Callable Statement 繼承 PreparedStatement。使用下列 Statement 物件來執行不同的 SQL 陳述式：

- 『Statement 介面』：執行沒有參數的簡單 SQL 陳述式。
- 第 69 頁的『PreparedStatement 介面』 - 執行經過前置編譯但不一定有 IN 參數的 SQL 陳述式。
- 第 61 頁的『CallableStatement 介面』 - 執行資料庫儲存程序的呼叫。 CallableStatement 不一定有 IN、OUT 與 INOUT 參數。

Statement 物件可讓您以單一群組的方式，透過批次支援的使用，對資料庫提出多重 SQL 指令。您可以使用批次支援來提升效能，因爲處理一組作業通常比一次處理一個作業還要快。關於使用批次支援的詳細資訊，請參閱 JDBC 支援的加強功能。

使用批次更新時，通常會關閉自動確定。關閉自動確定可讓您決定若發生錯誤及並未執行所有指令時，是否要確定異動。在 JDBC 2.0 與以上的版本 的 JDBC 規格中，Statement 物件記錄可順利以一個群組來提出及執行的指令清單。當 executeBatch() 方法執行批次指令的這個清單時，將依指令新增到清單的次序，來執行它們。

AS400JDBCStatement 提供的方法可讓您執行許多動作，它們包括：

- 執行不同種類的陳述式
- 擷取 Statement 物件不同的參數值，包括：
  - 連線
  - 執行 Statement 所建立的任何自動產生之鍵值
  - 提取大小和提取方向
  - 最大欄位大小和最大列數限制
  - 現行結果集、下一個結果集、結果集類型、結果集並行以及結果集游標保留性
- 新增 SQL 陳述式到現行批次中
- 執行 SQL 陳述式的現行批次

Statement 介面

使用 Connection.createStatement() 來建立新的 Statement 物件。

下面範例說明如何使用 Statement 物件。

```
                    // Connect to the server.
Connection c = DriverManager.getConnection("jdbc:as400://mySystem");

                    // Create a Statement object.
Statement s = c.createStatement();

                    // Run an SQL statement that creates
                    // a table in the database.
s.executeUpdate("CREATE TABLE MYLIBRARY.MYTABLE (NAME VARCHAR(20), ID INTEGER)");

                    // Run an SQL statement that inserts
                    // a record into the table.
```

```
s.executeUpdate("INSERT INTO MYLIBRARY.MYTABLE (NAME, ID) VALUES ('DAVE', 123)");

                  // Run an SQL statement that inserts
                  // a record into the table.
s.executeUpdate("INSERT INTO MYLIBRARY.MYTABLE (NAME, ID) VALUES ('CINDY', 456)");

                  // Run an SQL query on the table.
ResultSet rs = s.executeQuery("SELECT * FROM MYLIBRARY.MYTABLE");

                  // Close the Statement and the
                  // Connection.
s.close();
c.close();
```

JDBC XA 分散式異動管理:

JDBC XA 分散式異動管理類別可讓您在分散式異動中使用 IBM Toolbox for Java JDBC 驅動程式。使用 XA 類別可讓 IBM Toolbox for Java JDBC 驅動程式參與跨越多重資料來源的異動。

通常，XA 分散式異動管理類別是由異動管理程式 (與 JDBC 驅動程式分離) 所直接使用與控制。分散式異動管理介面定義爲 JDBC 2.0 Optional Package 及 Java Transaction API (JTA) 的一部分。 兩者皆由 Sun 以 jar 檔案的形式提供。JDBC 3.0 API 也支援分散式異動管理介面，此 API 隨附於 Java 2 Platform 標準版 1.4 版。

如需相關資訊，請參閱 Sun 網站的 JDBC 及 JTA 。

您可使用下列物件，讓 IBM Toolbox for Java JDBC 驅動程式參與 XA 分散式異動：

- AS400JDBCXADataSource - AS400JDBCXAConnection 物件的 Factory。 此爲 AS400JDBCDataSource 的次類別。
- AS400JDBCXACConnection - 已置於儲存區的連線物件，提供連線儲存區管理及 XA 資源管理的連結鉤。
- AS400JDBCXAResource - 用於 XA 異動管理的資源管理程式。

註: 在 V5R3 之前，資料庫主電腦伺服器使用工作範圍鎖定的 XA API (XA 模型)。在 V5R3 及後續版本中，資料庫主電腦伺服器將異動範圍鎖定的 XA API (NTS 模型) 用於所有 MTS 功能。如需這些 API 不同之處的相關資訊，請參閱 XA API。

範例：使用 XA 類別

下列範例說明 XA 類別的簡單用法。請記住明細將由使用其它資料來源的作業填入。此類型的程式碼通常出現在異動管理程式之內。

```
// Create an XA data source for making the XA connection.
AS400JDBCXADataSource xaDataSource = new AS400JDBCXADataSource("myAS400");
xaDataSource.setUser("myUser");
xaDataSource.setPassword("myPasswd");

// Get an XAConnection and get the associated XAResource.
// This provides access to the resource manager.
XAConnection xaConnection = xaDataSource.getXAConnection();
XAResource xaResource = xaConnection.getXAResource();

// Generate a new Xid (this is up to the transaction manager).
Xid xid = ...;

// Start the transaction.
xaResource.start(xid, XAResource.TMNOFLAGS);

// ...Do some work with the database...

// End the transaction.
```

```
        xaResource.end(xid, XAResource.TMSUCCESS);

        // Prepare for a commit.
        xaResource.prepare(xid);

        // Commit the transaction.
        xaResource.commit(xid, false);

        // Close the XA connection when done.  This implicitly
    // closes the XA resource.
        xaConnection.close();
```

Jobs 類別

IBM Toolbox for Java Jobs 類別 (在存取套件中) 可讓 Java 程式擷取及變更工作資訊。

註: Toolbox for Java 同時具備資源類別，可提供同屬組織架構及一致的程式設計介面，用以處理不同的 iSeries 物件與清單。在您閱讀過存取套件和 resource 套件中各種類別的相關資訊後，即可針對您的應用程式選擇最適合的物件。處理工作的資源類別爲 RJob、RJobList 及 RJobLog。

請將 Jobs 類別與下列類型的工作資訊搭配使用：

- 日期及時間資訊
- 工作佇列
- 語言識別字
- 記載的訊息
- 輸出佇列
- 印表機資訊

存取套件中的 Job 類別如下：

- Job - 擷取及變更 iSeries 工作資訊
- JobList - 擷取 iSeries 工作清單
- JobLog - 代表 iSeries 的工作日誌

範例

下列範例說明一些使用 Job、JobList 和 JobLog 類別的方法。第一個範例說明搭配使用快取記憶體與 Job 類別的方法。鏈接至緊接在範例程式碼之後的其他範例。

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

範例：在設定及取得某個值時使用快取記憶體

```
    try {

        // Creates AS400 object.
        AS400 as400 = new AS400("systemName");

        // Constructs a Job object
        Job job = new Job(as400,"QDEV002");

        // Gets job information
        System.out.println("User of this job :" + job.getUser());
        System.out.println("CPU used :" + job.getCPUUsed();
        System.out.println("Job enter system date : " + job.getJobEnterSystemDate());

        // Sets cache mode
```

```
            job.setCacheChanges(true);

            // Changes will be store in the cache.
            job.setRunPriority(66);
            job.setDateFormat("*YMD");

            // Commit changes. This will change the value on the iSeries.
     job.commitChanges();

            // Set job information to system directly(without cache).
            job.setCacheChanges(false);
            job.setRunPriority(60);
    } catch (Exception e)
  {
            System.out.println(quot;error :" + e)
  }
```

下列範例說明，如何列出屬於某位使用者的工作、列出含有工作狀態資訊的工作，以及顯示工作日誌中的訊息：

第 455 頁的『範例：使用 JobList 以列出工作識別資訊』

第 457 頁的『範例：使用 JobList 以取得工作清單』

第 460 頁的『範例：使用 JobLog 以顯示工作日誌中的訊息』

Job 類別:

Job 類別 (在存取套件中) 可讓 Java 程式擷取及變更伺服器工作資訊。

註: IBM Toolbox for Java 也具有資源類別，可提供同屬組織架構和一致的程式設計介面，用以處理 iSeries 伺服器上不同的物件及清單。在您閱讀過存取套件和 resource 套件中各種類別的相關資訊後，即可針對您的應用程式選擇最適合的物件。處理工作的資源類別爲 RJob、RJobList 及 RJobLog。

您可以透過 Job 類別擷取及變更下列類型的工作資訊：

- 工作佇列
- 輸出佇列
- 訊息日誌
- 印表機裝置
- 國家或地區 ID
- 日期格式

Job 類別也能使您一次變更一個單一值，或使用 setCacheChanges(true) 方法快取數個變更，並使用 commitChanges() 方法確定變更。如果未開啓快取，您將需要執行確定。

範例

如需程式碼範例，請參閱 Job 類別的 Javadoc 參考文件。此範例說明如何設定及取得快取記憶體中的值，以便使用 setRunPriority() 方法來設定執行優先順序，使用 setDateFormat() 方法設定日期格式：

Job

JobList 類別:

您可以使用 JobList 類別 (在存取套件中) 來列出 iSeries 工作。

註: IBM Toolbox for Java 同時具有資源類別，可提供同屬組織架構及一致的程式設計介面，用以處理不同的 iSeries 物件及清單。在您閱讀過存取套件和 resource 套件中各種類別的相關資訊後，即可針對您的應用程式選擇最適合的物件。處理工作的資源類別爲 RJob、RJobList 及 RJobLog。

透過 JobList 類別，您可以擷取下列：

- 所有工作
- 依名稱、工作號碼或使用者的工作

使用 getJobs() 方法傳回 iSeries 工作清單，或使用 getLength() 方法傳回最後一個 getJobs() 擷取的工作數。

範例：使用 JobList

下面範例會列出系統中所有作用中的工作：

```
                    // Create an AS400 object. List the
                    // jobs on this iSeries.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create the job list object.
JobList jobList = new JobList(sys);

                    // Get the list of active jobs.
Enumeration list = jobList.getJobs();

                    // For each active job on the system
                    // print job information.
while (list.hasMoreElements())
{
    Job j = (Job) list.nextElement();

    System.out.println(j.getName() + "." +
                       j.getUser() + "." +
                       j.getNumber());
}
```

JobLog 類別:

JobLog 類別 (在存取套件中) 以呼叫 getMessages() 的方式來擷取伺服器工作的工作日誌中的訊息。

註: IBM Toolbox for Java 同時具有資源類別，可提供同屬組織架構及一致的程式設計介面，用以處理不同的 iSeries 物件及清單。在您閱讀過存取套件和 resource 套件中各種類別的相關資訊後，即可針對您的應用程式選擇最適合的物件。處理工作的資源類別爲 RJob、RJobList 及 RJobLog。

範例：使用 JobLog

下面範例列印了指定的使用者在工作日誌中所有的相關訊息：

```
                    // ... Setup work to create an AS400
                    // object and a jobList object has
                    // already been done

                    // Get the list of active jobs on
                    // the iSeries
Enumeration list = jobList.getJobs();

                    // Look through the list to find a
                    // job for the specified user.
while (list.hasMoreElements())
{
   Job j = (Job) list.nextElement();
```

```
        if (j.getUser().trim().equalsIgnoreCase(userID))
        {
                       // A job matching the current user
                       // was found. Create a job log
                       // object for this job.
           JobLog jlog = new JobLog(system, j.getName(), j.getUser(), j.getNumber());

                       // Enumerate the messages in the job
                       // log then print them.
           Enumeration messageList = jlog.getMessages();

           while (messageList.hasMoreElements())
           {
               AS400Message message = (AS400Message) messageList.nextElement();
               System.out.println(message.getText());
           }

        }
    }
```

Message 類別

AS400Message

AS400Message 物件可讓 Java 程式擷取前一個作業 (如指令呼叫) 所產生的 i5/OS 訊息。從訊息物件中，Java 程式可擷取下列項目：

- 包含訊息的 iSeries 檔案庫及訊息檔案
- 訊息 ID
- 訊息類型
- 訊息嚴重性
- 訊息文字
- 訊息解說文字

下列範例說明如何使用 AS400Message 物件：

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
                       // Create a command call object.
    CommandCall cmd = new CommandCall(sys, "myCommand");

                       // Run the command
    cmd.run();

                       // Get the list of messages that are
                       // the result of the command that I
                       // just ran
    AS400Message[] messageList = cmd.getMessageList();

                      // Iterate through the list
                      // displaying the messages
    for (int i = 0; i < messageList.length; i++)
    {
       System.out.println(messageList[i].getText());
    }
```

範例：使用訊息清單

下列範例告訴您如何利用 CommandCall 和 ProgramCall 來使用訊息清單。

- 第 421 頁的『範例：使用 CommandCall』

- 第 472 頁的『範例：使用 ProgramCall』

QueuedMessage

QueuedMessage 類別延伸 AS400Message 類別。

註: Toolbox for Java 同時具備資源類別，可提供同屬組織架構及一致的程式設計介面，用以處理不同的 iSeries 物件與清單。在您閱讀過存取套件和 resource 套件中各種類別的相關資訊後，即可針對您的應用程式選擇最適合的物件。處理佇列訊息的資源類別爲 RQueuedMessage。

QueuedMessage 類別可針對 iSeries 訊息佇列存取訊息的相關資訊。透過此類別，Java 程式可以擷取：

- 關於訊息源起的資訊，如程式、工作名稱、工作號碼和使用者
- 訊息佇列
- 訊息金鑰
- 訊息回答狀態

下列範例列印現行 (已登入) 使用者的訊息佇列中的所有訊息：

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
                // The message queue is on this iSeries.
AS400 sys = new AS400(mySystem.myCompany.com);

                // Create the message queue object.
                // This object will represent the
                // queue for the current user.
MessageQueue queue = new MessageQueue(sys, MessageQueue.CURRENT);

                // Get the list of messages currently
                // in this user's queue.
Enumeration e = queue.getMessages();

                // Print each message in the queue.
while (e.hasMoreElements())
{
   QueuedMessage msg = e.getNextElement();
   System.out.println(msg.getText());
}
```

MessageFile

MessageFile 類別可讓您接收來自 iSeries 訊息檔的訊息。MessageFile 類別會傳回一個含有訊息的 AS400Message 物件。使用 MessageFile 類別，您可以執行下列：

- 傳回包含訊息的訊息物件
- 傳回包含訊息中的替代文字的訊息物件

下列範例說明如何擷取及列印訊息：

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
AS400 system = new AS400("mysystem.mycompany.com");
MessageFile messageFile = new MessageFile(system);
messageFile.setPath("/QSYS.LIB/QCPFMSG.MSGF");
AS400Message message = messageFile.getMessage("CPD0170");
System.out.println(message.getText());
```

MessageQueue

MessageQueue 類別可讓 Java 程式與 iSeries 訊息佇列相互作用。

註: Toolbox for Java 同時具備資源類別，可提供同屬組織架構及一致的程式設計介面，用以處理不同的 iSeries 物件與清單。在您閱讀過存取套件和 resource 套件中各種類別的相關資訊後，即可針對您的應用程式選擇最適合的物件。處理訊息佇列的資源類別爲 RMessageQueue。

MessageQueue 類別作爲 QueuedMessage 類別的儲存區。 尤其，getMessages() 方法傳回 QueuedMessage 物件清單。MessageQueue 類別可執行下列：

- 設定訊息佇列屬性
- 取得關於訊息佇列的資訊
- 接收訊息佇列中的訊息
- 傳送訊息到訊息佇列
- 回答訊息

下列範例列出現行使用者的訊息佇列中的訊息：

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
                  // The message queue is on this iSeries.
AS400 sys = new AS400(mySystem.myCompany.com);

                  // Create the message queue object.
                  // This object will represent the
                  // queue for the current user.
MessageQueue queue = new MessageQueue(sys, MessageQueue.CURRENT);

                  // Get the list of messages currently
                  // in this user's queue.
Enumeration e = queue.getMessages();

                  // Print each message in the queue.
while (e.hasMoreElements())
{
   QueuedMessage msg = e.getNextElement();
   System.out.println(msg.getText());
}
```

NetServer

NetServer 已棄用並由類別 ISeriesNetServer 置換。

NetServer 類別代表 iSeries 伺服器上的 NetServer 服務。NetServer 物件可讓您查詢及修改 NetServer 服務的狀態及配置。

例如，您可以使用 NetServer 類別來：

- 啓動或停止 NetServer
- 取得所有目前檔案共用及列印共用的清單
- 取得所有現行階段作業的清單
- 查詢及變更屬性值 (使用繼承自 ChangeableResource 的方法)

註: 若要使用 NetServer 類別，您必須要有具備 *IOSYSCFG 權限的伺服器使用者設定檔。

NetServer 類別爲 ChangeableResource 及 Resource 的延伸，因此其提供了代表不同 NetServer 值及設定值的「屬性」集合。您可查詢或變更屬性，來存取或變更 NetServer 的配置。部分 NetServer 的屬性包括：

- NAME
- NAME_PENDING
- DOMAIN
- ALLOW_SYSTEM_NAME
- AUTOSTART
- CCSID
- WINS_PRIMARY_ADDRESS

擱置中屬性

許多 NetServer 屬性爲擱置中屬性 (例如， NAME_PENDING)。擱置中屬性代表下一次您在伺服器啓動 (或重新啓動) NetServer 時，才會生效的 NetServer 值。

當您有一對相關的屬性，其中一個屬性爲擱置中屬性，而另一個屬性爲非擱置中屬性時：

- 擱置中屬性的狀態會爲讀取/寫入，因此可加以變更
- 非擱置中屬性屬於唯讀屬性，因此只能查詢，不能加以變更

其他 NetServer 類別

相關的 NetServer 類別可讓您取得及設定特定連線、階段作業、檔案共用及列印共用的相關詳細資訊：

- NetServerConnection：代表 NetServer 連線
- NetServerFileShare：代表 NetServer 檔案伺服器共用
- NetServerPrintShare：代表 NetServer 列印伺服器共用
- NetServerSession：代表 NetServer 階段作業
- NetServerShare：代表 NetServer 共用

範例：使用 NetServer 物件變更 NetServer 的名稱

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
        // Create a system object to represent the iSeries server.
        AS400 system = new AS400("MYSYSTEM", "MYUSERID", "MYPASSWD");

        // Create an object with which to query and modify the NetServer.
        NetServer nServer = new NetServer(system);

        // Set the "pending name" to NEWNAME.
        nServer.setAttributeValue(NetServer.NAME_PENDING, "NEWNAME");

        // Commit the changes.  This sends the changes to the server.
    nServer.commitAttributeChanges();

        // The NetServer name will get set to NEWNAME the next time the NetServer
        // is ended and started.
```

許可權類別

許可權類別可讓您取得及設定物件權限資訊。物件權限資訊也稱爲許可權。「許可權」類別代表許多使用者對特定物件的權限的集合。UserPermission 類別代表單一使用者對特定物件的權限。

許可權類別

Permission 類別可讓您擷取及變更物件權限資訊。它包括許多有權使用物件的使用者的集合。Permission 物件容許 Java 程式快取權限變更，直到呼叫 commit() 方法爲止。一旦呼叫了 commit() 方法，則到那時候所做的變更都會傳送到伺服器。「許可權」類別提供的一些功能如下：

- addAuthorizedUser()：新增授權使用者。
- commit()：向伺服器確定許可權變更。
- getAuthorizationList()：傳回物件的授權清單。
- getAuthorizedUsers()：傳回授權使用者的列舉。
- getOwner()：傳回物件擁有者的名稱。
- getSensitivityLevel()：傳回物件的靈敏度層次。
- getType()：傳回物件權限類型 (QDLO、QSYS 或 Root)。
- getUserPermission()：傳回特定使用者的許可權給物件。
- getUserPermissions()：傳回使用者的許可權列舉給物件。
- setAuthorizationList()：設定物件的授權清單。
- setSensitivityLevel()：設定物件的靈敏度層次。

範例：使用許可權

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

下列範例將說明如何建立一個許可權，以及如何在物件中新增授權使用者。

```
// Create AS400 object
AS400 as400 = new AS400();

    // Create Permission passing in the AS400 and object
    Permission myPermission = new Permission(as400, "QSYS.LIB/myLib.LIB");

    // Add a user to be authorized to the object
    myPermission.addAuthorizedUser("User1");
```

UserPermission 類別

UserPermission 類別代表單一特定使用者的權限。UserPermission 具有三種次類別，它們依據物件類型來處理權限：

UserPermission 類別	說明
DLOPermission	代表使用者對儲存在 QDLS 中的「文件檔案庫物件 (DLO)」的權限。
QSYSPermission	代表使用者對儲存在 QSYS.LIB 及包含在伺服器中的物件的權限。
RootPermission	代表使用對包含在主目錄結構中的物件的權限。RootPermissions 物件即是那些未包含在 QSYS.LIB 或 QDLS 的物件。

UserPermission 類別可讓您執行下列：

- 判斷使用者設定檔是否爲群組設定檔
- 傳回使用者設定檔名稱

- 指出使用者是否具有權限
- 爲授權清單管理設定權限

範例：使用 UserPermission

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

下列範例將說明，如何擷取對物件具有許可權的使用者和群組，並將他們逐一列印出來。

```
            // Create a system object.
            AS400 sys = new AS400("MYAS400", "USERID", "PASSWORD");

            // Represent the permissions to an object on the system, such as a library.
            Permission objectInQSYS = new Permission(sys, "/QSYS.LIB/FRED.LIB");

            // Retrieve the various users/groups that have permissions set on that object.
            Enumeration enum = objectInQSYS.getUserPermissions();
       while
(enum.hasMoreElements ())
      {
                // Print out the user/group profile names one at a time.
                UserPermission userPerm = (UserPermission)enum.nextElement();
                System.out.println(userPerm.getUserID());
      }
```

DLOPermission 類別:

DLOPermission UserPermission 的次類別。DLOPermission 可讓您顯示及設定使用者對文件檔案庫物件 (DLO) 具有的權限 (稱爲許可權)。

下列其中一個權限將指定給每一使用者。

權限值	說明
*ALL	使用者可執行所有的作業，除了授權清單管理所控制的作業以外。
*AUTL	授權清單係用來決定文件的權限。
*CHANGE	使用者可以對物件變更及執行基本的功能。
*EXCLUDE	使用者無法存取物件。
*USE	使用者有物件作業權限、讀取權限和執行權限。

您必須使用下列方法之一，方能變更或決定使用者的權限：

- 使用 getDataAuthority() 來顯示使用者的權限值
- 使用 setDataAuthority() 來設定使用者的權限值

在設定許可權後，務必要從 Permissions 類別中使用 commit() 方法，將變更傳送至伺服器。

如需許可權及權限的相關資訊，請參閱 **iSeries Security Reference**  的 Chapter 5: Resource Security。

範例：使用 DLOPermission

下列範例說明如何擷取及列印 DLO 許可權，包括每個許可權的使用者設定檔。

```
            // Create a system object.

            AS400 sys = new AS400("MYAS400", "USERID", "PASSWORD");
```

```
            // Represent the permissions to a DLO object.
      Permission objectInQDLS = new Permission(sys, "/QDLS/MyFolder");

            // Print the object pathname and retrieve its permissions.
      System.out.println("Permissions on " + objectInQDLS.getObjectPath() + " are as follows:");
            Enumeration enum = objectInQDLS.getUserPermissions();
       while
(enum.hasMoreElements ())
      {
                // For each of the permissions, print out the user profile name
                // and that user's authorities to the object.
                DLOPermission dloPerm = (DLOPermission)enum.nextElement();
        System.out.println(dloPerm.getUserID() + ": " + dloPerm.getDataAuthority());
      }
```

QSYSPermission:

QSYSPermission 爲 UserPermission 類別的次類別。對於儲存在 QSYS.LIB 內的傳統 iSeries 檔案庫結構中的物件，您可以使用 QSYSPermission 來顯示及設定其使用者所具備的許可權。儲存在 QSYS.LIB 中的物件，可經由設定系統定義的權限值、或設定個別物件權限、或設定個別物件及資料權限，來設定它的權限。

下表列出並說明系統定義的有效權限值：

系統定義的權限值	說明
*ALL	使用者可以執行所有作業，但那些由授權清單管理所控制的作業除外。
*AUTL	授權清單係用來決定文件的權限。
*CHANGE	使用者可以對物件變更及執行基本的功能。
*EXCLUDE	使用者無法存取物件。
*USE	使用者有物件作業權限、讀取權限和執行權限。

每個系統定義的權限值實際上代表個別物件權限與資料權限的組合。下表說明個別物件的系統定義權限與資料權限之間的關係：

表 1. **Y** *指出可以指定的權限。n 指出不能指定的權限。*

系統定義的權限	物件權限					資料權限				
	Opr	**Mgt**	**Exist**	**Alter**	**Ref**	**Read**	**Add**	**Upd**	**Dlt**	**Exe**
全部	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**
變更	**Y**	n	n	n	n	**Y**	**Y**	**Y**	**Y**	**Y**
排除	n	n	n	n	n	n	n	n	n	n
使用	**Y**	n	n	n	n	**Y**	n	n	n	**Y**
Autl	限定於使用者 (*PUBLIC) 和決定個別物件和資料權限的指定授權清單才有效。									

指定系統定義的權限時，即自動指派適當的個別權限。同樣地，指定各種個別權限時，即變更適當的個別權限值。當個別的物件權限與資料權限的組合不對映單一的系統定義權限值時，該單一值則變成「使用者定義」。

請使用 getObjectAuthority() 方法來顯示現行的系統定義權限。 請使用 setObjectAuthority() 方法，透過單一值設定現行的系統定義權限。

使用適當的 set 方法，將個別物件權限值設定爲 on 或 off：

- setAlter()

- setExistence()
- setManagement()
- setOperational()
- setReference()

使用適當的 set 方法，將個別資料權限值設定爲 on 或 off：

- setAdd()
- setDelete()
- setExecute()
- setRead()
- setUpdate()

如需不同權限的相關資訊，請參閱 **iSeries Security Reference**  的 Chapter 5: Resource Security。如需使用 iSeries CL 指令來授予及編輯物件權限的相關資訊，請參閱 iSeries CL 指令授予物件權限 (GRTOBJAUT) 及編輯物件權限 (EDTOBJAUT)。

範例

這個範例告訴您如何擷取及列印 QSYS 物件的許可權。

```
            // Create a system object.
            AS400 sys = new AS400("MYAS400", "USERID", "PASSWORD");

            // Represent the permissions to a QSYS object.
            Permission objectInQSYS = new Permission(sys, "/QSYS.LIB/FRED.LIB");

            // Print the object pathname and retrieve its permissions.
            System.out.println("Permissions on "+objectInQSYS.getObjectPath()+" are as follows:");
            Enumeration enum = objectInQSYS.getUserPermissions();
       while
(enum.hasMoreElements ())
      {
                // For each of the permissions, print out the user profile name
                // and that user's authorities to the object.
                QSYSPermission qsysPerm = (QSYSPermission)enum.nextElement();
                System.out.println(qsysPerm.getUserID()+": "+qsysPerm.getObjectAuthority());
      }
```

RootPermission:

RootPermission 爲 UserPermission 類別的次類別。RootPermission 類別可讓您顯示及設定使用者對包含在主目錄結構中的物件的許可權。

主目錄結構上的物件可以設定資料權限或物件權限。您可以把資料權限設定爲下表所列的值。請使用 getDataAuthority() 方法來顯示現行值，使用 setDataAuthority() 方法來設定資料權限。

下表列出並說明有效資料權限值：

資料權限值	說明
*none	使用者沒有物件的權限。
*RWX	使用者具有讀取、新增、更新、刪除與執行權限。
*RW	使用者具有讀取、新增與刪除權限。
*RX	使用者具有讀取及執行權限。

資料權限值	說明
*WX	使用者具有新增、更新、刪除與執行權限。
*R	使用者具有讀取權限。
*W	使用者具有新增、更新與刪除權限。
*X	使用者具有執行權限。
*EXCLUDE	使用者無法存取物件。
*AUTL	這個物件上的公用權限來自於授權清單。

這些物件權限可設定爲下列一個或多個值：更新、存在、管理或參照。您可以使用 setAlter()、setExistence()、setManagement() 或 setReference() 方法來設定值爲開啓或關閉。

設定了物件的資料權限或物件權限之後，請務必從 Permissions 類別使用 commit() 方法，將變更傳送到伺服器。

如需不同權限的相關資訊，請參閱 **iSeries Security Reference**  的 Chapter 5: Resource Security。

範例

這個範例告訴您如何擷取及列印主物件的的許可權。

```
// Create a system object.
AS400 sys = new AS400("MYAS400", "USERID", "PASSWORD");

// Represent the permissions to an object in the root file system.
Permission objectInRoot = new Permission(sys, "/fred");

// Print the object pathname and retrieve its permissions.
System.out.println("Permissions on "+objectInRoot.getObjectPath()+" are as follows:");
Enumeration enum = objectInRoot.getUserPermissions();
while (enum.hasMoreElements())
{
  // For each of the permissions, print out the user profile name
  // and that user's authorities to the object.
  RootPermission rootPerm = (RootPermission)enum.nextElement();
  System.out.println(rootPerm.getUserID()+": "+rootPerm.getDataAuthority());
}
```

Print 類別

列印物件包括排存檔、輸出佇列、印表機、印表機檔案、寫出器工作及「進階功能列印™ (AFP)」資源，這些包括字型、格式定義、套印格式、頁面定義與頁面區段。只有在「版本 3 版次 7 (V3R7)」及更新版本的 i5/OS 系統中，才可存取 AFP 資源。(嘗試對執行比 V3R7 還早的版本的系統開啓 AFPResourceList 將產生 RequestNotSupportedException 異常情況。)

IBM Toolbox for Java 的列印物件類別是依據基礎類別 PrintObject 及六個列印物件類型每一個的次類別所組織而成的。基礎類別包含所有伺服器列印物件共用的方法與屬性。次類別包含專屬每一個次類型使用的方法與屬性。

在下列工作中使用 print 類別：

- 使用伺服器列印物件：
  - PrintObjectList 類別 - 用於列示及使用伺服器列印物件。(列印物件包括排存檔、輸出佇列、印表機、「進階功能列印 (AFP)」資源、印表機檔案及寫出器工作)
  - PrintObject 基礎類別 - 供使用列印物件使用

- 擷取 PrintObject 屬性
- 使用 SpooledFileOutputStream 類別建立新的伺服器排存檔 (用於 EBCDIC 爲主的印表機資料)
- 產生 SNA 字元串流 (SCS) 印表機資料串流
- 使用 PrintObjectInputStream 讀取排存檔及 AFP 資源
- 讀取排存檔，利用 PrintObjectPageInputStream
- 複製排存檔
- 檢視「進階功能列印 (AFP)」以及「SNA 字元串流 (SCS)」排存檔

範例

- 範例：建立排存檔說明如何從輸入串流中建立伺服器的排存檔
- 範例：建立 SCS 排存檔說明如何使用 SCS3812Writer 類別產生 SCS 資料串流，以及如何將串流寫入伺服器的排存檔中
- 範例：讀取排存檔說明如何使用 PrintObjectInputStream 來讀取現有的伺服器排存檔
- 範例：讀取及轉換排存檔說明如何使用 PrintObjectPageInputStream 及 PrintObjectTransformedInputStream 在讀取排存檔時獲得不同的轉換
- 範例：複製排存檔說明如何將排存檔複製到您要複製的檔案所在的相同佇列上。
- 範例：(使用接聽程式) 非同步列出排存檔說明如何以非同步方式列出系統中的所有排存檔，以及在建置清單時，如何使用 PrintObjectListListener 介面取得回饋
- 範例：(不使用接聽程式) 非同步列出排存檔說明如何在不使用 PrintObjectListListener 介面的情況下，以非同步方式列出系統中的所有排存檔
- 範例：同步列出排存檔說明如何以同步方式列出系統中的所有排存檔

列出 Print 物件:

您可以使用 PrintObjectList 類別及其次類別來使用列印物件的清單。每一個次類別均具有一些方法，以便容許依據對特殊類型的 print 物件有意義的 條件，來過濾清單。例如 SpooledFileList 容許您依據排存檔的建立者、排存檔所在的輸出佇列、紙張規格或排存檔的使用者資料，來過濾排存檔的清單。僅有那些符合過濾條件的排存檔才會列示出來。如果未設定任何過濾條件，將使用每一個過濾條件的預設值。

若要從伺服器實際擷取列印物件的清單，請使用 openSynchronously() 或 openAsynchronously() 方法。除非已從伺服器擷取清單中的所有物件，否則不會傳回 openSynchronously() 方法。openAsynchronously() 方法將會立即傳回，而且在等待清單的建立時，呼叫程式可以在前景中執行其它工作。非同步開啓的清單同時也容許呼叫程式在物件傳回時，對使用者顯示這些物件。因爲當物件傳回時，使用者可以看到它們，所以在使用者看來，回應時間似乎更快。事實上，就整體而言，回應時間可能更長，因爲會對清單中的每一個物件做額外的處理。

如果清單是以非同步方式開啓，則呼叫程式可在清單建立時取得回應。例如 isCompleted() 和 size() 等方法會指出清單是否已經建立完成，或是傳回清單目前的大小。其它方法例如 waitForListToComplete() 和 waitForItem() 則容許呼叫程式等待清單完成，或等待一個特殊項目。除了呼叫這些 PrintObjectList 方法之外，呼叫程式也可以透過清單登錄爲聆聽者。如有這種情況，將通知呼叫程式在清單中所發生的事件。若要登記事件或取消事件的登記，呼叫程式會使用 PrintObjectListListener()，然後再呼叫 addPrintObjectListListener() 來登記或呼叫 removePrintObjectListListener() 來取消登記。下表將顯示從 PrintObjectList 傳遞的事件。

PrintObjectList 事件	事件傳遞時
listClosed	清單結束時。
listCompleted	清單完成時。
listErrorOccurred	如果擷取清單時有丟出任何異常情況時。

PrintObjectList 事件	事件傳遞時
listOpened	清單開啓時。
listObjectAdded	在清單中新增物件時。

在清單開啓並且清單中的物件已處理過之後，請使用 close() 方法來關閉清單。這將釋出在開啓期間，已配置給垃圾收集器的任何資源。關閉清單之後，可修改它的過濾條件，然後可再重新開啓該清單。

列出列印物件後，伺服器會傳送所列出的每一個列印物件的屬性，並隨著列印物件一起儲存。更新這些屬性可以使用 PrintObject 類別中的 update() 方法。伺服器傳回哪些屬性，可根據列出的列印物件類型來決定。有關各類型之列印物件的屬性有一份預設清單存在，想要置換時可使用 PrintObjectList 中的 setAttributesToRetrieve() 方法。請參閱擷取 PrintObject 屬性章節，取得各類型 print 物件支援的屬性清單。

只有在「版本 3 版次 7」及更新版次的 i5/OS 中，才可列出「AFP 資源」。 對 V3R7 之前的舊版系統開啓 AFPResourceList 會產生 RequestNotSupportedException 的異常。

範例

下列範例顯示各種列出排存檔的方式。

第 466 頁的『範例：非同步列出排存檔 (使用接收程式)』 說明如何以非同步方式列出系統中的所有排存檔，以及在建置清單時，如何使用 PrintObjectListListener 介面取得回饋

第 469 頁的『範例：非同步列出排存檔 (不使用接收程式)』 說明如何在不使用 PrintObjectListListener 介面的情況下，以非同步方式列出系統中的所有排存檔。

第 471 頁的『範例：同步列出排存檔』 說明如何以同步方式列出系統中的所有排存檔

使用 Print 物件:

PrintObject 是一種抽象類別。(抽象類別不容許您建立類別的實例。相反地，您必須建立其次類別之一的實例。) 使用任何下列方法來建立次類別的物件：

- 如果知道系統和物件的識別屬性，請呼叫物件的公用建構子來明確地建構物件。
- 您可以使用 PrintObjectList 次類別來建置物件清單，然後從清單中取得個別物件。
- 由於所呼叫的某方法或 set 方法，結果會建立並傳回某物件。例如，在 WriterJob 類別中的 靜態方法 start() 會傳回 WriterJob 物件。

請使用基本類別、PrintObject 及其次類別來使用伺服器列印物件：

- OutputQueue
- 印表機
- PrinterFile
- SpooledFile
- WriterJob

擷取 PrintObject 屬性:

您可以使用屬性 ID 以及來自基礎 PrintObject 類別的其中一個方法來擷取列印物件屬性：

- 請使用 getIntegerAttribute(int attributeID) 來擷取整數類型屬性。
- 請使用 getFloatAttribute(int attributeID) 來擷取浮點類型屬性。

- 請使用 getStringAttribute(int attributeID) 來擷取字串類型屬性。

attributeID 參數是一個識別要擷取哪個屬性的整數。在基礎 PrintObject 類別中全部 ID 都定義成公用常數。PrintAttributes 檔包含每一個屬性 ID 的登錄。登錄包括屬性說明和它的類型 (整數、浮點或字串)。若要列示可能使用這些方法所擷取的屬性，請選取下列鏈接：

- 用於 AFP 資源的 AFPResourceAttrs
- 用於輸出佇列的 OutputQueueAttrs
- 用於印表機的 PrinterAttrs
- 用於印表機檔案的 PrinterFileAttrs
- 用於排存檔的 SpooledFileAttrs
- 用於寫出器工作的 WriterJobAttrs

若要達到可接受的效能，這些屬性會複製到用戶端中。如果隱含地建立物件，則在列示物件時或第一次需要物件時會複製這些屬性。此動作可在每次應用程式需要擷取屬性時，防止將物件傳送到主電腦。此動作也有可能使 Java 列印物件實例包含關於伺服器物件的過期資訊。物件的使用者可對物件呼叫 update() 方法來復新全部屬性。此外，如果應用程式對物件呼叫任何會變更物件屬性的方法，則會自動更新屬性。例如，若輸出佇列有 RELEASED 的狀態屬性 (getStringAttribute(ATTR_OUTQSTS)；傳回字串 RELEASED)， 且 hold() 方法在輸出佇列中被呼叫，則取得狀態屬性，之後傳回 HELD。

setAttributes 方法

您可以使用 setAttributes 方法來變更排存檔及印表機檔案物件的屬性。選取下列鏈接取得清單，或選取可設定的屬性：

- 用於印表機檔案的 PrinterFileAttrs
- 用於排存檔的 SpooledFileAttrs

setAttributes 方法取用 PrintParameterList 參數，這是用來保留屬性 ID 及其值的集合的類別。清單原本是空的，呼叫程式可以對它使用不同的 setParameter() 方法，新增屬性到清單中。

PrintParameterList 類別

您可以使用 PrintParameterList 類別來傳送某屬性群組到某個使用許多屬性作爲參數的方法。例如，您可以使用 SpooledFile 方法 sendTCP()，傳送使用使用 TCP (LPR) 的排存檔。PrintParameterList 物件包含 send 指令的必要參數 (如遠程系統與佇列) 以及想要的任何選用參數，如傳送排存檔之後是否要刪除排存檔。在這些情況中，方法文件提供必要與選用屬性的清單。PrintParameterList setParameter() 方法不會檢查設定哪些屬性及對屬性設定哪些值。PrintParameterList setParameter() 方法僅包含要傳送至此方法的值。一般而言，系統不處理 PrintParameterList 中的額外屬性，而且對屬性使用的不合法值會在伺服器中加以診斷。

AFP 資源屬性:

擷取屬性

使用適當的 getIntegerAttribute()、getStringAttribute() 或 getFloatAttribute() 方法，可爲 AFP 資源擷取下列屬性：

- ATTR_AFP_RESOURCE - AFP 資源整合檔案系統路徑
- ATTR_OBJEXTATTR - 物件延伸屬性
- ATTR_DESCRIPTION - 本文說明
- ATTR_DATE - 開啓檔案的日期
- ATTR_TIME - 開啓檔案的時間

- ATTR_NUMBYTES - 要讀取/寫入的位元組數

設定屬性

不允許針對 AFP 資源設定屬性。

輸出佇列屬性:

擷取屬性

使用適當的 getIntegerAttribute()、getStringAttribute() 或 getFloatAttribute() 方法，可爲輸出佇列擷取下列屬性：

- ATTR_AUTHCHCK - 要檢查的權限
- ATTR_DATA_QUEUE - 資料佇列整合檔案系統名稱
- ATTR_DISPLAYANY - 顯示任何檔案
- ATTR_JOBSEPRATR - 工作分隔字元
- ATTR_NUMFILES - 檔案數
- ATTR_NUMWRITERS - 佇列上已啓動的寫出器數量
- ATTR_OPCNTRL - 操作員控制的
- ATTR_ORDER - 佇列上的檔案次序
- ATTR_OUTPUT_QUEUE - 輸出佇列整合檔案系統名稱
- ATTR_OUTQSTS - 輸出佇列狀態
- ATTR_PRINTER - 印表機
- ATTR_SEPPAGE - 分隔頁
- ATTR_DESCRIPTION - 本文說明
- ATTR_USRDEFOPT - 使用者定義選項
- ATTR_USER_DEFINED_OBJECT - 使用者定義物件整合檔案系統名稱
- ATTR_USER_TRANSFORM_PROG - 使用者轉換程式整合檔案系統名稱
- ATTR_USER_DRIVER_PROG - 使用者驅動程式整合檔案系統名稱
- ATTR_WTRJOBNAME - 寫出器工作名稱
- ATTR_WTRJOBNUM - 寫出器工作編號
- ATTR_WTRJOBSTS - 寫出器工作狀態
- ATTR_WTRJOBUSER - 寫出器工作使用者名稱

設定屬性

不可針對輸出佇列設定的屬性。

印表機屬性:

擷取屬性

使用適當的 getIntegerAttribute()、getStringAttribute() 或 getFloatAttribute() 方法，可爲印表機擷取下列屬性：

- ATTR_AFP - 進階功能列印
- ATTR_ALIGNFORMS - 對齊格式
- ATTR_ALWDRTPRT - 允許直接列印
- ATTR_BTWNCPYSTS - 份數間狀態

- ATTR_BTWNFILESTS - 檔案間狀態
- ATTR_CODEPAGE - 字碼頁
- ATTR_CHANGES - 變更
- ATTR_DEVCLASS - 裝置類別
- ATTR_DEVMODEL - 裝置模型
- ATTR_DEVTYPE - 裝置類型
- ATTR_DEVSTATUS - 裝置狀態
- ATTR_DRWRSEP - 分隔頁送紙匣
- ATTR_ENDPNDSTS - 結束擱置中狀態
- ATTR_FILESEP - 檔案分隔字元
- ATTR_FONTID - 字型 ID
- ATTR_FORM_DEFINITION - 套表定義整合檔案系統名稱
- ATTR_FORMTYPE - 紙張規格
- ATTR_FORMTYPEMSG - 紙張規格訊息
- ATTR_FORMFEED - 換頁
- ATTR_CHAR_ID - 圖形字集
- ATTR_HELDSTS - 已保留狀態
- ATTR_HOLDPNDSTS - 保留擱置中狀態
- ATTR_JOBUSER - 工作使用者
- ATTR_MFGTYPE - 製造商類型及模型
- ATTR_MESSAGE_QUEUE - 訊息佇列整合檔案系統名稱
- ATTR_ONJOBQSTS - 工作中佇列狀態
- ATTR_OUTPUT_QUEUE - 輸出佇列整合檔案系統名稱
- ATTR_OVERALLSTS - 整體狀態
- ATTR_POINTSIZE - 字體大小
- ATTR_PRINTER - 印表機
- ATTR_PRTDEVTYPE - 印表機裝置類型
- ATTR_PUBINF_COLOR_SUP - 支援的發佈資訊顏色
- ATTR_PUBINF_PPM_COLOR - 每分鐘發佈資訊頁數 (彩色)
- ATTR_PUBINF_PPM - 每分鐘發佈資訊頁數 (單色)
- ATTR_PUBINF_DUPLEX_SUP - 發佈資訊雙工支援
- ATTR_PUBINF_LOCATION - 發佈資訊位置
- ATTR_RMTLOCNAME - 遠端位置名稱
- ATTR_SPOOLFILE - 排存檔名稱
- ATTR_SPLFNUM - 排存檔號碼
- ATTR_STARTEDBY - 由使用者啓動
- ATTR_DESCRIPTION - 本文說明
- ATTR_USERDATA - 使用者資料
- ATTR_USRDEFOPT - 使用者定義選項
- ATTR_USER_DEFINED_OBJECT - 使用者定義物件整合檔案系統名稱

- ATTR_USER_TRANSFORM_PROG - 使用者轉換程式整合檔案系統名稱
- ATTR_USER_DRIVER_PROG - 使用者驅動程式整合檔案系統名稱
- ATTR_SCS2ASCII - 轉換 SCS 爲 ASCII
- ATTR_WTNGDATASTS - 等待資料狀態
- ATTR_WTNGDEVSTS - 等待裝置狀態
- ATTR_WTNGMSGSTS - 等待訊息狀態
- ATTR_WTRAUTOEND - 自動結束寫出器的時機
- ATTR_WTRJOBNAME - 寫出器工作名稱
- ATTR_WTRJOBSTS - 寫出器工作狀態
- ATTR_WTRSTRTD - 已啓動寫出器
- ATTR_WRTNGSTS - 寫入狀態

設定屬性

不可針對印表機設定的屬性。

印表機檔案屬性:

擷取屬性

使用適當的 getIntegerAttribute()、getStringAttribute() 或 getFloatAttribute() 方法， 可爲印表機檔案擷取下列屬性：

- ATTR_ALIGN - 調整頁面
- ATTR_BKMGN_ACR - 底面邊距橫向偏移
- ATTR_BKMGN_DWN - 底面邊距向下偏移
- ATTR_BACK_OVERLAY - 底面套板整合檔案系統名稱
- ATTR_BKOVL_DWN - 底面套板向下偏移
- ATTR_BKOVL_ACR - 底面套板橫向偏移
- ATTR_CPI - 每吋字元數
- ATTR_CODEDFNTLIB - 編碼字型檔案庫名稱
- ATTR_CODEPAGE - 字碼頁
- ATTR_CODEDFNT - 編碼字型名稱
- ATTR_CONTROLCHAR - 控制字元
- ATTR_CONVERT_LINEDATA - 轉換行式資料
- ATTR_COPIES - 份數
- ATTR_CORNER_STAPLE - 裝訂角
- ATTR_DBCSDATA - 使用者指定的 DBCS 資料
- ATTR_DBCSEXTENSN - DBCS 延伸字元
- ATTR_DBCSROTATE - DBCS 字元旋轉
- ATTR_DBCSCPI - 每吋 DBCS 字元數
- ATTR_DBCSSISO - DBCS SO/SI 間距
- ATTR_DFR_WRITE - 延遲寫入
- ATTR_PAGRTT - 頁旋轉度

- ATTR_EDGESTITCH_NUMSTAPLES - 邊緣裝訂針數
- ATTR_EDGESTITCH_REF - 邊緣裝訂參照
- ATTR_EDGESTITCH_REFOFF - 邊緣裝訂參照
- ATTR_ENDPAGE - 結束頁
- ATTR_FILESEP - 檔案分隔字元
- ATTR_FOLDREC - 摺疊記錄
- ATTR_FONTID - 字型 ID
- ATTR_FORM_DEFINITION - 套表定義整合檔案系統名稱
- ATTR_FORMFEED - 換頁
- ATTR_FORMTYPE - 紙張規格
- ATTR_FTMGN_ACR - 正面邊距橫向偏移
- ATTR_FTMGN_DWN - 正面邊距向下偏移
- ATTR_FRONT_OVERLAY - 正面套板整合檔案系統名稱
- ATTR_FTOVL_ACR - 正面套板橫向偏移
- ATTR_FTOVL_DWN - 正面套板向下偏移
- ATTR_CHAR_ID - 圖形字集
- ATTR_JUSTIFY - 硬體調整
- ATTR_HOLD - 保留排存檔
- ATTR_LPI - 每吋行數
- ATTR_MAXRCDS - 最大排存輸出記錄
- ATTR_OUTPTY - 輸出優先順序
- ATTR_OUTPUT_QUEUE - 輸出佇列整合檔案系統名稱
- ATTR_OVERFLOW - 溢位行號
- ATTR_PAGE_DEFINITION - 頁面定義整合檔案系統
- ATTR_PAGELEN - 頁長度
- ATTR_MEASMETHOD - 測量方法
- ATTR_PAGEWIDTH - 頁寬度
- ATTR_MULTIUP - 每邊的頁數
- ATTR_POINTSIZE - 字體大小
- ATTR_FIDELITY - 列印精確度
- ATTR_DUPLEX - 雙面列印
- ATTR_PRTQUALITY - 列印品質
- ATTR_PRTTEXT - 列印文字
- ATTR_PRINTER - 印表機
- ATTR_PRTDEVTYPE - 印表機裝置類型
- ATTR_RPLUNPRT - 置換不可列印的字元
- ATTR_RPLCHAR - 置換字元
- ATTR_SADDLESTITCH_NUMSTAPLES - 騎縫裝訂針數
- ATTR_SADDLESTITCH_REF - 騎縫裝訂參照
- ATTR_SAVE - 儲存排存檔

- ATTR_SRCDRWR - 送紙匣
- ATTR_SPOOL - 排存資料
- ATTR_SCHEDULE - 排存的輸出時程表
- ATTR_STARTPAGE - 起始頁
- ATTR_DESCRIPTION - 本文說明
- ATTR_UNITOFMEAS - 測量單位
- ATTR_USERDATA - 使用者資料
- ATTR_USRDEFDATA - 使用者定義資料
- ATTR_USRDEFOPT - 使用者定義選項
- ATTR_USER_DEFINED_OBJECT - 使用者定義物件整合檔案系統名稱

設定屬性

使用 setAttributes() 方法設定印表機檔案的下列屬性：

- ATTR_ALIGN - 調整頁面
- ATTR_BKMGN_ACR - 底面邊距橫向偏移
- ATTR_BKMGN_DWN - 底面邊距向下偏移
- ATTR_BACK_OVERLAY - 底面套板整合檔案系統名稱
- ATTR_BKOVL_DWN - 底面套板向下偏移
- ATTR_BKOVL_ACR - 底面套板橫向偏移
- ATTR_CPI - 每吋字元數
- ATTR_CODEDFNTLIB - 編碼字型檔案庫名稱
- ATTR_CODEPAGE - 字碼頁
- ATTR_CODEDFNT - 編碼字型名稱
- ATTR_CONTROLCHAR - 控制字元
- ATTR_CONVERT_LINEDATA - 轉換行式資料
- ATTR_COPIES - 份數
- ATTR_CORNER_STAPLE - 裝訂角
- ATTR_DBCSDATA - 使用者指定的 DBCS 資料
- ATTR_DBCSEXTENSN - DBCS 延伸字元
- ATTR_DBCSROTATE - DBCS 字元旋轉
- ATTR_DBCSCPI - 每吋 DBCS 字元數
- ATTR_DBCSSISO - DBCS SO/SI 間距
- ATTR_DFR_WRITE - 延遲寫入
- ATTR_PAGRTT - 頁旋轉度
- ATTR_EDGESTITCH_NUMSTAPLES - 邊緣裝訂針數
- ATTR_EDGESTITCH_REF - 邊緣裝訂參照
- ATTR_EDGESTITCH_REFOFF - 邊緣裝訂參照
- ATTR_ENDPAGE - 結束頁
- ATTR_FILESEP - 檔案分隔字元
- ATTR_FOLDREC - 摺疊記錄

- ATTR_FONTID - 字型 ID
- ATTR_FORM_DEFINITION - 套表定義整合檔案系統名稱
- ATTR_FORMFEED - 換頁
- ATTR_FORMTYPE - 紙張規格
- ATTR_FTMGN_ACR - 正面邊距橫向偏移
- ATTR_FTMGN_DWN - 正面邊距向下偏移
- ATTR_FRONT_OVERLAY - 正面套板整合檔案系統名稱
- ATTR_FTOVL_ACR - 正面套板橫向偏移
- ATTR_FTOVL_DWN - 正面套板向下偏移
- ATTR_CHAR_ID - 圖形字集
- ATTR_JUSTIFY - 硬體調整
- ATTR_HOLD - 保留排存檔
- ATTR_LPI - 每吋行數
- ATTR_MAXRCDS - 最大排存輸出記錄
- ATTR_OUTPTY - 輸出優先順序
- ATTR_OUTPUT_QUEUE - 輸出佇列整合檔案系統名稱
- ATTR_OVERFLOW - 溢位行號
- ATTR_PAGE_DEFINITION - 頁面定義整合檔案系統
- ATTR_PAGELEN - 頁長度
- ATTR_MEASMETHOD - 測量方法
- ATTR_PAGEWIDTH - 頁寬度
- ATTR_MULTIUP - 每邊的頁數
- ATTR_POINTSIZE - 字體大小
- ATTR_FIDELITY - 列印精確度
- ATTR_DUPLEX - 雙面列印
- ATTR_PRTQUALITY - 列印品質
- ATTR_PRTTEXT - 列印文字
- ATTR_PRINTER - 印表機
- ATTR_PRTDEVTYPE - 印表機裝置類型
- ATTR_RPLUNPRT - 置換不可列印的字元
- ATTR_RPLCHAR - 置換字元
- ATTR_SADDLESTITCH_NUMSTAPLES - 騎縫裝訂針數
- ATTR_SADDLESTITCH_REF - 騎縫裝訂參照
- ATTR_SAVE - 儲存排存檔
- ATTR_SRCDRWR - 送紙匣
- ATTR_SPOOL - 排存資料
- ATTR_SCHEDULE - 排存的輸出時程表
- ATTR_STARTPAGE - 起始頁
- ATTR_DESCRIPTION - 本文說明
- ATTR_UNITOFMEAS - 測量單位

- ATTR_USERDATA - 使用者資料
- ATTR_USRDEFDATA - 使用者定義資料
- ATTR_USRDEFOPT - 使用者定義選項
- ATTR_USER_DEFINED_OBJECT - 使用者定義物件整合檔案系統名稱

排存檔屬性:

擷取屬性

使用適當的 getIntegerAttribute()、getStringAttribute() 或 getFloatAttribute() 方法， 可爲排存檔擷取下列屬性：

- ATTR_AFP - 進階功能列印
- ATTR_ALIGN - 調整頁面
- ATTR_BKMGN_ACR - 底面套板橫向偏移
- ATTR_BKMGN_DWN - 底面套板向下偏移
- ATTR_BACK_OVERLAY - 底面套板整合檔案系統名稱
- ATTR_BKOVL_DWN - 底面套板向下偏移
- ATTR_BKOVL_ACR - 底面套板橫向偏移
- ATTR_CPI - 每吋字元數
- ATTR_CODEDFNTLIB - 編碼字型檔案庫名稱
- ATTR_CODEDFNT - 編碼字型名稱
- ATTR_CODEPAGE - 字碼頁
- ATTR_CONTROLCHAR - 控制字元
- ATTR_COPIES - 份數
- ATTR_COPIESLEFT - 待產生份數
- ATTR_CORNER_STAPLE - 裝訂角
- ATTR_CURPAGE - 現行頁
- ATTR_DATE - 物件建立日期
- ATTR_DATE_WTR_BEGAN_FILE - 寫出器開始處理排存檔日期
- ATTR_DATE_WTR_CMPL_FILE - 寫出器完成處理排存檔日期
- ATTR_DBCSDATA - 使用者指定的 DBCS 資料
- ATTR_DBCSEXTENSN - DBCS 延伸字元
- ATTR_DBCSROTATE - DBCS 字元旋轉
- ATTR_DBCSCPI - 每吋 DBCS 字元數
- ATTR_DBCSSISO - DBCS SO/SI 間距
- ATTR_PAGRTT - 頁旋轉度
- ATTR_EDGESTITCH_NUMSTAPLES - 邊緣裝訂針數
- ATTR_EDGESTITCH_REF - 邊緣裝訂參照
- ATTR_EDGESTITCH_REFOFF - 邊緣裝訂參照偏移
- ATTR_ENDPAGE - 結束頁
- ATTR_FILESEP - 檔案分隔字元
- ATTR_FOLDREC - 摺疊記錄

- ATTR_FONTID - 字型 ID
- ATTR_FORM_DEFINITION - 套表定義整合檔案系統名稱
- ATTR_FORMFEED - 換頁
- ATTR_FORMTYPE - 紙張規格
- ATTR_FTMGN_ACR - 正面邊距橫向偏移
- ATTR_FTMGN_DWN - 正面邊距向下偏移
- ATTR_FRONTSIDE_OVERLAY - 正面套板整合檔案系統名稱
- ATTR_FTOVL_ACR - 正面套板橫向偏移
- ATTR_FTOVL_DWN - 正面套板向下偏移
- ATTR_CHAR_ID - 圖形字集
- ATTR_JUSTIFY - 硬體調整
- ATTR_HOLD - 保留排存檔
- ATTR_IPP_ATTR_CHARSET - IPP 屬性字集
- ATTR_IPP_JOB_ID - IPP 工作 ID
- ATTR_IPP_JOB_NAME - IPP 工作名稱
- ATTR_IPP_JOB_NAME_NL - IPP 工作名稱 NL
- ATTR_IPP_JOB_ORIGUSER - 產生 IPP 工作的使用者
- ATTR_IPP_JOB_ORIGUSER_NL - 產生 IPP 工作的使用者 NL
- ATTR_IPP_PRINTER_NAME - IPP 印表機名稱
- ATTR_JOBNAME - 工作名稱
- ATTR_JOBNUMBER - 工作編號
- ATTR_JOBUSER - 工作使用者
- ATTR_JOB_SYSTEM - 工作系統
- ATTR_LASTPAGE - 列印末頁
- ATTR_LINESPACING - 行距
- ATTR_LPI - 每吋行數
- ATTR_MAXRCDS - 最大排存輸出記錄
- ATTR_PAGELEN - 頁長度
- ATTR_PAGEWIDTH - 頁寬度
- ATTR_MEASMETHOD - 測量方法
- ATTR_NETWORK - 網路 ID
- ATTR_NUMBYTES - 讀取/寫入的位元組數
- ATTR_OUTPUTBIN - 輸出紙匣
- ATTR_OUTPTY - 輸出優先順序
- ATTR_OUTPUT_QUEUE - 輸出佇列整合檔案系統名稱
- ATTR_OVERFLOW - 溢位行號
- ATTR_MULTIUP - 每邊的頁數
- ATTR_POINTSIZE - 字體大小
- ATTR_FIDELITY - 列印精確度
- ATTR_DUPLEX - 雙面列印

- ATTR_PRTQUALITY - 列印品質
- ATTR_PRTTEXT - 列印文字
- ATTR_PRINTER - 印表機
- ATTR_PRTASSIGNED - 已指定印表機
- ATTR_PRTDEVTYPE - 印表機裝置類型
- ATTR_PRINTER_FILE - 印表機檔案整合檔案系統名稱
- ATTR_RECLENGTH - 記錄長度
- ATTR_REDUCE - 減少輸出
- ATTR_RPLUNPRT - 置換不可列印的字元
- ATTR_RPLCHAR - 置換字元
- ATTR_RESTART - 重新啓動列印
- ATTR_SADDLESTITCH_NUMSTAPLES - 騎縫裝訂針數
- ATTR_SADDLESTITCH_REF - 騎縫裝訂參照
- ATTR_SAVE - 儲存排存檔
- ATTR_SRCDRWR - 送紙匣
- ATTR_SPOOLFILE - 排存檔名稱
- ATTR_SPLFNUM - 排存檔號碼
- ATTR_SPLFSTATUS - 排存檔狀態
- ATTR_SCHEDULE - 排存的輸出時程表
- ATTR_STARTPAGE - 起始頁
- ATTR_SYSTEM - 系統建立位置
- ATTR_TIME - 物件建立時間
- ATTR_TIME_WTR_BEGAN_FILE - 寫出器開始處理排存檔時間
- ATTR_TIME_WTR_CMPL_FILE - 寫出器完成處理排存檔時間
- ATTR_PAGES - 總頁數
- ATTR_UNITOFMEAS - 測量單位
- ATTR_USERCMT - 使用者註解
- ATTR_USERDATA - 使用者資料
- ATTR_USRDEFDATA - 使用者定義資料
- ATTR_USRDEFFILE - 使用者定義檔
- ATTR_USRDEFOPT - 使用者定義選項
- ATTR_USER_DEFINED_OBJECT - 使用者定義物件整合檔案系統名稱

設定屬性

使用 setAttributes() 方法設定排存檔的下列屬性：

- ATTR_ALIGN - 調整頁面
- ATTR_BACK_OVERLAY - 底面套板整合檔案系統名稱
- ATTR_BKOVL_DWN - 底面套板向下偏移
- ATTR_BKOVL_ACR - 底面套板橫向偏移
- ATTR_COPIES - 份數

- ATTR_ENDPAGE - 結束頁
- ATTR_FILESEP - 檔案分隔字元
- ATTR_FORM_DEFINITION - 套表定義整合檔案系統名稱
- ATTR_FORMFEED - 換頁
- ATTR_FORMTYPE - 紙張規格
- ATTR_FRONTSIDE_OVERLAY - 正面套板整合檔案系統名稱
- ATTR_FTOVL_ACR - 正面套板橫向偏移
- ATTR_FTOVL_DWN - 正面套板向下偏移
- ATTR_OUTPTY - 輸出優先順序
- ATTR_OUTPUT_QUEUE - 輸出佇列整合檔案系統名稱
- ATTR_MULTIUP - 每邊的頁數
- ATTR_FIDELITY - 列印精確度
- ATTR_DUPLEX - 雙面列印
- ATTR_PRTQUALITY - 列印品質
- ATTR_PRTSEQUENCE - 列印順序
- ATTR_PRINTER - 印表機
- ATTR_RESTART - 重新啓動列印
- ATTR_SAVE - 儲存排存檔
- ATTR_SCHEDULE - 排存的輸出時程表
- ATTR_STARTPAGE - 起始頁
- ATTR_USERDATA - 使用者資料
- ATTR_USRDEFOPT - 使用者定義選項
- ATTR_USER_DEFINED_OBJECT - 使用者定義物件整合檔案系統名稱

寫出器工作屬性:

擷取屬性

使用適當的 getIntegerAttribute()、getStringAttribute() 或 getFloatAttribute() 方法，可爲寫出器工作擷取下列屬性：

- ATTR_WTRJOBNAME - 寫出器工作名稱
- ATTR_WTRJOBNUM - 寫出器工作編號
- ATTR_WTRJOBSTS - 寫出器工作狀態
- ATTR_WTRJOBUSER - 寫出器工作使用者名稱

設定屬性

不可針對寫出器工作設定的屬性。

列印物件屬性:

目錄

- 對齊格式
- 調整頁面
- 允許直接列印
- 權限
- 檢查權限
- 自動結束寫出器
- 輔助儲存體
- 底面邊距橫向位移
- 底面邊距向下位移
- 背底套板
- 底面套板橫向位移
- 底面套板向下位移
- 份數間狀態
- 檔案間狀態
- 變更
- 每吋字元數
- 字碼頁
- 字碼字型名稱
- 編碼字型檔案庫名稱
- 控制字元
- 轉換行式資料
- 份數
- 待產生份數
- 裝訂角
- 現行頁
- 資料格式
- 資料佇列
- 開啓檔案日期
- 排存檔工作建立結束日期
- 寫出器開始處理排存檔日期
- 寫出器完成處理排存檔日期
- 使用者指定的 DBCS 資料
- DBCS (雙位元組字集) 延伸字元
- DBCS (雙位元組字集) 字元旋轉
- DBCS (雙位元組字集) 每吋字元數
- DBCS (雙位元組字集) SO/SI 間距
- 延遲寫入
- 頁旋轉度
- 傳送後刪除檔案
- 目的地選項

- 目的地類型
- 裝置類別
- 裝置模型
- 裝置狀態
- 裝置類型
- 顯示任何檔案
- 分隔頁送紙匣
- 裝訂邊縫線數
- 邊縫線參照
- 邊縫線參照位移
- 結束擱置中狀態
- 結束頁
- 訊息封來源
- 檔案分隔字元
- 摺疊記錄
- 字型 ID
- 套表定義
- 換頁
- 紙張規格
- 紙張規格訊息選項
- 正面邊距橫向位移
- 正面邊距向下位移
- 正面套板
- 正面套板橫向位移
- 正面套板向下位移
- 圖形字集
- 硬體調整
- 已保留狀態
- 保留排存檔
- 保留擱置中狀態
- 影像配置
- 起始設定寫出器
- 網際網路位址
- IPP 屬性字集
- IPP 工作 ID
- IPP 工作名稱
- IPP 工作名稱 NL
- IPP 工作產生的使用者名稱
- IPP 工作產生的使用者名稱 NL
- IPP 印表機名稱

- 工作名稱
- 工作編號
- 工作分隔頁
- 工作系統
- 工作使用者
- 列印末頁
- 頁長度
- 檔案庫名稱
- 每吋行數
- 行距
- 製造商類型及模型
- 每一用戶端清單最大工作數
- 最大排存輸出記錄
- 測量方法
- 訊息說明
- 訊息 ID
- 訊息佇列
- 訊息回答
- 訊息文字
- 訊息類型
- 訊息嚴重性
- 多重項目回答功能
- 網路 ID
- 網路列印伺服器物件屬性
- 排存檔中的位元組數
- 讀取/寫入位元組數
- 檔案數
- 佇列上已啓動寫出器的數量
- 物件延伸屬性
- 工作中佇列狀態
- 開啓時間指令
- 操作員已控制
- 佇列上的檔案次序
- 輸出紙匣
- 輸出優先順序
- 輸出佇列
- 輸出佇列狀態
- 整體狀態
- 溢位行號
- 一次一頁

- 預估頁數
- 頁面定義
- 頁次
- 每邊的頁數
- 紙張來源 1
- 紙張來源 2
- 圖素密度
- 字體大小
- 列印清晰度
- 雙面列印
- 列印品質
- 列印順序
- 列印文字
- 印表機
- 已指定印表機
- 印表機裝置類型
- 印表機檔案
- 印表機佇列
- 支援的發佈資訊顏色
- 每分鐘發佈資訊頁數 (彩色)
- 每分鐘發佈資訊頁數 (單色)
- 發佈資訊雙工支援
- 發佈資訊位置
- 遠端位置名稱
- 記錄長度
- 減少輸出
- 遠端系統
- 置換不可列印的字元
- 置換字元
- 重新啓動列印
- 馬鞍型裝訂縫針數
- 馬鞍型縫針參照
- 儲存排存檔
- 探查位移
- 探查原點
- 傳送優先順序
- 分隔頁
- 來源匣
- 排存 SCS
- 排存資料

- 排存檔建立鑑別方法
- 排存檔建立安全性方法
- 排存檔名稱
- 排存檔號碼
- 排存檔狀態
- 排存輸出時程表
- 由使用者啓動
- 起始頁
- 系統建立位置
- 本文說明
- 開啓檔案時間
- 排存檔工作建立結束時間
- 寫出器開始處理排存檔時間
- 寫出器完成處理排存檔時間
- 總頁數
- 轉換 SCS 爲 ASCII
- 測量單位
- 使用者註解
- 使用者資料
- 使用者定義資料
- 使用者定義檔案
- 使用者定義物件
- 使用者定義選項
- 使用者驅動程式資料
- 使用者驅動程式
- 使用者 ID
- 使用者 ID 位址
- 使用者轉換程式
- 檢視清晰度
- VM/MVS 類別
- 等待資料狀態
- 等待裝置狀態
- 等待訊息狀態
- 自動結束寫出器時間
- 結束寫出器時間
- 保留檔案時間
- 頁寬度
- 工作站自訂物件
- 寫出器工作名稱
- 寫出器工作編號

- 寫出器工作狀態
- 寫出器工作使用者名稱
- 寫出器已啓動
- 寫出器起始頁
- 寫入狀態
- NPS CCSID
- NPS 層次

進階功能列印

ID ATTR_AFP

類型 字串

說明 指出此排存檔是否使用排存檔外部的 AFP 資源。有效值爲 *YES 及 *NO。

AFP 資源

ID ATTR_AFP_RESOURCE

類型 字串

說明 外部 AFP (進階功能列印) 資源的「整合檔案系統」路徑。「整合檔案系統」路徑的格式爲 /QSYS.LIB/library.LIB/resource.type，其中 *library* 是指包含資源的檔案庫， *resource* 是指資源的名稱，而 *type* 則是指資源類型。*type* 的有效值包括：FNTRSC、FORMDF、OVL、PAGSEG 及 PAGDFN。

對齊格式

ID ATTR_ALIGNFORMS

類型 字串

說明 傳送格式對齊訊息的時間。有效值爲 *WTR、*FILE 及 *FIRST。

調整頁面

ID ATTR_ALIGN

類型 字串

說明 指出列印此排存檔之前是否先傳送格式調整訊息。有效值爲 *YES 及 *NO。

允許直接列印

ID ATTR_ALWDRTPRT

類型 字串

說明 指出印表機寫出器是否可將印表機配置給直接列印到印表機的工作。有效值爲 *YES 及 *NO。

權限

ID ATTR_AUT

類型 字串

說明 指定賦予使用者的權限，該使用者不具有對輸出佇列的特定權限。有效值爲 *USE、*ALL、*CHANGE、*EXCLUDE 及 *LIBCRTAUT。

檢查權限

ID ATTR_AUTCHK

類型 字串

說明 指出輸出佇列可讓使用者在輸出佇列上控制所有檔案的權限類型。有效值爲 *OWNER 及 *DTAAUT。

自動結束寫出器

ID ATTR_AUTOEND

類型 字串

說明 指定是否必須自動結束寫出器。有效值爲 *NO 及 *YES。

輔助儲存體

ID ATTR_AUX_POOL

類型 整數

說明 指定儲存排存檔的輔助儲存體儲存區 (ASP) 數量。 可能的值爲：

- 1：系統 ASP
- 2-32：其中一個使用者 ASP

底面邊距橫向位移

ID ATTR_BACKMGN_ACR

類型 浮點

說明 針對紙張的背面，若有需要，可指定從左邊開始列印的距離。特殊值 *FRONTMGN 將編碼爲 -1。

底面邊距向下位移

ID ATTR_BACKMGN_DWN

類型 浮點

說明 針對紙張的背面，若有需要，可指定從頂端開始列印的距離。特殊值 *FRONTMGN 將編碼爲 -1。

底面套板

ID ATTR_BACK_OVERLAY

類型 字串

說明 底面套板的「整合檔案系統」路徑或特殊值。如果值是「整合檔案系統」路徑，則其格式爲 /QSYS.LIB/library.LIB/overlay.OVL， 其中 *library* 是資源的檔案庫，而 *overlay* 則是底板的名稱。有效的特殊值包括 *FRONTOVL。

底面套板橫向位移

ID ATTR_BKOVL_ACR

類型 浮點

說明 從列印套板之原點的橫向位移。

底面套板向下位移

ID ATTR_BKOVL_DWN

類型 浮點

說明 從列印套板之原點向下偏移。

份數間狀態

ID ATTR_BTWNCPYSTS

類型 字串

說明 寫出器是否介於多重副本排存檔的份數之間。回覆值爲 *YES 或 *NO。

檔案間狀態

ID ATTR_BTWNFILESTS

類型 字串

說明 寫出器是否介於檔案之間。回覆值爲 *YES 或 *NO。

變更

ID ATTR_CHANGES

類型 字串

說明 擱置中變更生效的時間。有效值爲 *NORDYF、*FILEEND，或是指未對寫出器變更擱置的空白。

每吋字元數

ID ATTR_CPI

類型 浮點

說明 每一水平英吋的字元數。

字碼頁

ID ATTR_CODEPAGE

類型 字串

說明 此排存檔之字碼點的圖形字元對映。若圖形字元集欄位中有特殊值，則此欄位可能包括 0。

字碼字型名稱

ID ATTR_CODEDFNT

類型 字串

說明 編碼字型名稱。編碼字型是一種 AFP 資源，由字集及字碼頁所組成。特殊值包括 *FNTCHRSET。

編碼字型檔案庫名稱

ID ATTR_CODEDFNTLIB

類型 字串

說明 包含編碼字型的檔案庫名稱。如果編碼字型名稱有特殊值，則此欄位可能包括空白。

控制字元

ID ATTR_CONTROLCHAR

類型 字串

說明 此檔案是否使用「美國國際標準」印表機控制字元。可能的值爲 *NONE，表示沒有將列印控制字元傳入列印的資料中，或是 *FCFC，表示每一筆記錄的第一個字元都是「美國國際標準」印表機控制字元。

轉換行式資料

ID ATTR_CONVERT_LINEDATA

類型 字串

說明 行式資料在寫入排存前是否轉換爲 AFPDS。可能的值爲 *NO 及 *YES。

份數

ID ATTR_COPIES

類型 整數

說明 爲此排存檔所產生的總份數。

待產生份數

ID ATTR_COPIESLEFT

類型 整數

說明 爲此排存檔所產生的剩餘份數。

裝訂角

ID ATTR_CORNER_STAPLE

類型 字串

說明 裝訂角所使用的參照角落。裝訂在參照角落導入媒體。有效值爲 *NONE、*DEVD、*BOTRIGHT、*TOPRIGHT、*TOPLEFT 及 *BOTLEFT。

現行頁

ID ATTR_CURPAGE

類型 整數

說明 由寫出器工作寫入的現行頁。

資料格式

ID ATTR_DATAFORMAT

類型 字串

說明 資料格式。有效值爲 *RCDDATA 及 *ALLDATA。

資料佇列

ID ATTR_DATA_QUEUE

類型 字串

說明 指定與輸出佇列相關之資料佇列的「整合檔案系統」路徑，或是在沒有與輸出佇列相關的資料佇列時指定爲 *NONE。「整合檔案系統」路徑的格式爲「/QSYS.LIB/library.LIB/dataqueue.QTAQ」，其中 *library* 是包含資料佇列的檔案庫， 而 *dataqueue* 則是資料佇列的名稱。

開啓檔案日期

ID ATTR_DATE

類型 字串

說明 針對排存檔，則這是開啓此排存檔的日期。針對 AFP 資源，則是前次修改物件的日期。日期是以 C YY MM DD 的格式所編碼的字串。

排存檔工作建立結束日期

ID ATTR_DATE_END

類型 字串

說明 在系統上建立排存檔之工作的結束日期。如果「啓動排存檔建立日期」欄位設定爲 *ALL，則此欄位必須爲空白。如果已指定「啓動排存檔建立日期」欄位的日期，則應爲有效的日期。日期必須是 CYYMMDD 格式，或是下列其中一個特殊值：

- *LAST：要傳回的是建立日期及時間都等於或大於啓動的排存檔建立日期的所有排存檔。
- Date：要傳回的是建立日期及時間都等於或大於啓動的排存檔建立日期及時間，而小於或等於結束的排存檔日期及時間的所有排存檔。

日期格式 CYYMMDD 定義如下：

- C 爲世紀，其中 0 表示 19xx 年，而 1 表示 20xx 年
- YY 爲年份
- MM 爲月份
- DD 爲日

寫出器開始處理排存檔日期

ID ATTR_DATE_WTR_BEGAN_FILE

類型 字串

說明 指出寫出器開始處理此排存檔的日期。日期是以 C YY MM DD 的格式所編碼的字串。

寫出器完成處理排存檔日期

ID ATTR_DATE_WTR_CMPL_FILE

類型 字串

說明 指出寫出器處理完此排存檔的日期。日期是以 C YY MM DD 的格式所編碼的字串。

使用者指定的 DBCS 資料

ID ATTR_DBCSDATA

類型 字串

說明 排存檔是否包含雙位元組字集 (DBCS) 資料。有效值爲 *NO 及 *YES。

DBCS (雙位元組字集) 延伸字元

ID ATTR_DBCSEXTENSN

類型 字串

說明 系統是否要處理 DBCS (雙位元組字集) 延伸字元。有效值爲 *NO 及 *YES。

DBCS (雙位元組字集) 字元旋轉

ID ATTR_DBCAROTATE

類型 字串

說明 DBCS (雙位元組字集) 字元在列印前是否以逆時鐘方向 90 度旋轉。有效值爲 *NO 及 *YES。

DBCS (雙位元組字集) 每吋字元數

ID ATTR_DBCSCPI

類型 整數

說明 每英吋列印的雙位元組字元數。有效值爲 -1、-2、5、6 及 10。*CPI 的值已編碼爲 -1。*CONDENSED 的值已編碼爲 -2。

DBCS (雙位元組字集) SO/SI 間距

ID ATTR_DBCSSISO

類型 字串

說明 判定列印時移出及移入字元的表示法。有效值爲 *NO、*YES 及 *RIGHT。

延遲寫入

ID ATTR_DFR_WRITE

類型 字串

說明 列印前是否在系統緩衝區中保留列印資料。

頁旋轉度

ID ATTR_PAGRTT

類型 整數

說明 頁面上文字的旋轉度，與套表載入印表機的方向有關。有效值爲 -1、-2、-3、0、90、180 及 270。*AUTO 的值已編碼爲 -1、*DEVD 的值已編碼爲 -2 及 *COR 的值已編碼爲 -3。

傳送後刪除檔案

ID ATTR_DELETESPLF

類型 字串

說明 是否在傳送後刪除排存檔？有效值爲 *NO 及 *YES。

目的地選項

ID ATTR_DESTOPTION

類型 字串

說明 目的地選項。可讓使用者將選項傳給接收系統的字串。

目的地類型

ID ATTR_DESTINATION

類型 字串

說明 目的地類型。有效值爲 *OTHER、*AS400 及 *PSF2。

裝置類別

ID ATTR_DEVCLASS

類型 字串

說明 裝置的類別。

裝置模型

ID ATTR_DEVMODEL

類型 字串

說明 裝置的型號。

裝置狀態

ID ATTR_DEVSTATUS

類型 整數

說明 印表機裝置的狀態。有效值爲 0 (已轉斷)、10 (轉斷擱置)、20 (轉接擱置)、30 (已轉接)、40 (連線擱置)、60 (作用中)、66 (作用中寫出器)、70 (已保留)、75 (關閉電源)、80 (回復擱置)、90 (回復已取消)、100 (失敗)、106 (寫出器已失敗)、110 (處理中)、111 (已損壞)、112 (已鎖定) 及 113 (不明)。

裝置類型

ID ATTR_DEVTYPE

類型 字串

說明 裝置類型。

顯示任何檔案

ID ATTR_DISPLAYANY

類型 字串

說明 有權限讀取此輸出佇列的使用者是否能在此佇列上顯示任何輸出檔的輸出資料，或是僅能在其自有的檔案中顯示資料。有效值爲 *YES、*NO 及 *OWNER。

分隔頁送紙匣

ID ATTR_DRWRSEP

類型 整數

說明 識別要採用之工作及檔案分隔字元的匣。有效值爲 -1、-2、1、2 及 3。*FILE 值已編碼爲 -1， *DEVD 值已編碼爲 -2。

裝訂邊縫線數

ID ATTR_EDGESTITCH_NUMSTAPLES

類型 整數

說明 沿著完成的作業軸所套用的裝訂針數。

邊縫線參照

ID ATTR_EDGESTITCH_REF

類型 字串

說明 一或多個裝訂在沿著完成作業軸導向媒體之處。有效值爲 *NONE、*DEVD、*BOTTOM、*RIGHT、*TOP 及 *LEFT。

邊縫線參照位移

ID ATTR_EDGESTITCH_REFOFF

類型 浮點

說明 從參照邊到媒體中央之邊縫線的位移。

結束擱置中狀態

ID ATTR_ENDPNDSTS

類型 字串

說明 是否對此寫出器發出結束寫出器 (ENDWTR) 指令。可能的值爲 *NO - 不發出 ENDWTR 指令，*IMMED - 寫出器在其輸出緩衝區已清空時即結束， *CTRLD - 寫出器在目前的排存檔已列印後即結束，*PAGEED - 寫出器在頁尾結束。

結束頁

ID ATTR_ENDPAGE

類型 整數

說明 結束列印排存檔的頁次。有效值爲 0 或結束頁的頁次。*END 值已編碼爲 0。

訊息封來源

ID ATTR_ENVLP_SOURCE

類型 字串

說明 訊息封來源中的訊息封大小。若未指定此欄位或爲無效的值，則使用特殊值 *MFRTYPMDL。有效值爲 *NONE - 沒有訊息封來源、*MFRTYPMDL - 使用的製造商類型及模型所建議的訊息封大小、*MONARCH (3.875 x 7.5 吋)、*NUMBER9 (3.875 x 8.875 吋)、*NUMBER10 (4.125 x 9.5 吋)、*B5 (176mm x 250mm)、*C5 (162mm x 229mm) 及 *DL (110mm x 220mm)。

檔案分隔字元

ID ATTR_FILESEP

類型 整數

說明 置於排存檔每一份開頭的檔案分隔字元數。有效值爲 -1，或爲分隔字元的數量。*FILE 值已編碼爲 -1。

摺疊記錄

ID ATTR_FOLDREC

類型 字串

說明 超出列印用紙寬度的記錄是否要摺疊 (覆蓋) 到下一行。有效值爲 *YES 及 *NO。

字型 ID

ID ATTR_FONTID

類型 字串

說明 所使用的印表機字型。有效的特殊值包括 *CPI 及 *DEVD。

套表定義

ID ATTR_FORM_DEFINITION

類型 字串

說明 套表定義的「整合檔案系統」路徑名稱或特殊值。如果「整合檔案系統」路徑的格式爲「/QSYS.LIB/library.LIB/formdef.FORMDF」，其中 *library* 是套表定義的檔案庫，而 *formdef* 則是套表定義的名稱。有效的特殊值包括 *NONE、*INLINE、*INLINED 及 *DEVD。

換頁

ID ATTR_FORMFEED

類型 字串

說明 印表機的換頁方式。有效值爲 *CONT、*CUT、*AUTOCUT 及 *DEVD。

紙張規格

ID ATTR_FORMTYPE

類型 字串

說明 要列印此排存檔之印表機中所載入的套表類型。

紙張規格訊息選項

ID ATTR_FORMTYPEMSG

類型 字串

說明 完成現行套表類型時，傳送訊息給寫出器訊息佇列的訊息選項。有效值爲 *MSG、*NOMSG、*INFOMSG 及 *INQMSG。

正面邊距橫向位移

ID ATTR_FTMGN_ACR

類型 浮點

說明 針對紙張的正面，若有需要，可指定從左邊開始列印的距離。特殊值 *DEVD 已編碼爲 -2。

正面邊距向下位移

ID ATTR_FTMGN_DWN

類型 浮點

說明 針對紙張的正面，若有需要，可指定從頂端開始列印的距離。特殊值 *DEVD 已編碼爲 -2。

正面套板

ID ATTR_FRONT_OVERLAY

類型 字串

說明 正面套板的「整合檔案系統」路徑。「整合檔案系統」路徑的格式爲「/QSYS.LIB/library.LIB/overlay.OVL」，其中 *library* 是資源的檔案庫，而 *overlay* 則是套印格式的名稱。*NONE 字串用來表示未指定正面套板。

正面套板橫向位移

ID ATTR_FTOVL_ACR

類型 浮點

說明 從列印套板之原點的橫向位移。

正面套板向下位移

ID ATTR_FTOVL_DWN

類型 浮點

說明 從列印套板之原點向下偏移。

圖形字集

ID ATTR_CHAR_ID

類型 字串

說明 列印此檔案時使用的圖形字集。有效的特殊值包括 *DEVD、*SYSVAL 及 *JOBCCSID。

硬體調整

ID ATTR_JUSTIFY

類型 整數

說明 正確調整輸出的百分比。有效值爲 0、50 及 100。

已保留狀態

ID ATTR_HELDSTS

類型 字串

說明 是否保留寫出器。有效值爲 *YES 及 *NO。

保留排存檔

ID ATTR_HOLD

類型 字串

說明 是否保留排存檔。有效值爲 *YES 及 *NO。

保留擱置中狀態

ID ATTR_HOLDPNDSTS

類型 字串

說明 是否對此寫出器發出保留寫出器 (HLDWTR) 指令。可能的值爲 *NO - 不發出 HLDWTR 指令，*IMMED - 寫出器在其輸出緩衝區已清空時即保留，*CTRLD - 寫出器在目前的排存檔已列印後即保留，*PAGEED - 寫出器在頁尾保留。

影像配置

ID ATTR_IMGCFG

類型 字串

說明 影像變化的變換服務及列印資料串流格式。

起始設定寫出器

ID ATTR_WTRINIT

類型 字串

說明 使用者可指定何時要起始設定印表機裝置。有效值爲 *WTR、*FIRST 及 *ALL。

網際網路位址

ID ATTR_INTERNETADDR

類型 字串

說明 接收系統的網際網路位址。

IPP 屬性字集

ID ATTR_IPP_ATTR_CHARSET

類型 字串

說明 指出指定排存檔屬性之 IPP 的字集 (編碼字集及編碼方法)。

IPP 工作 ID

ID ATTR_IPP_JOB_ID

類型 整數

說明 與建立工作之 IPP 印表機相關的 IPP 工作 ID。

IPP 工作名稱

ID ATTR_IPP_ATR_CHARSET

類型 字串

說明 使用者好記的工作名稱。

IPP 工作名稱 NL

ID ATTR_IPP_JOB_NAME_NL

類型 字串

說明 工作名稱的自然語言。

IPP 工作產生的使用者名稱

ID ATTR_IPP_JOB_ORIGUSER

類型 字串

說明 定義提出此 IPP 工作的一般使用者。

IPP 工作產生的使用者名稱 NL

ID ATTR_IPP_JOB_ORIGUSER_NL

類型 字串

說明 定義工作產生之使用者名稱的自然語言。

IPP 印表機名稱

ID ATTR_IPP_PRINTER_NAME

類型 字串

說明 定義建立此工作的 IPP 印表機。

工作名稱

ID ATTR_JOBNAME

類型 字串

說明 建立排存檔的工作名稱。

工作編號

ID ATTR_JOBNUMBER

類型 字串

說明 建立排存檔的工作編號。

工作分隔頁

ID ATTR_JOBSEPRATR

類型 整數

說明 在此輸出佇列上，放置於每一個有排存檔的工作之輸出開頭的工作分隔頁數。有效值爲 -2 及 0-9。*MSG 的值已編碼爲 -2。在建立輸出佇列時指定工作分隔頁。

工作系統

ID ATTR_JOBSYSTEM

類型 字串

說明 執行建立排存檔的系統工作。

工作使用者

ID ATTR_JOBUSER

類型 字串

說明 建立排存檔之使用者名稱。

列印末頁

ID ATTR_LASTPAGE

類型 整數

說明 如果在工作完成處理之前結束列印，則列印末頁爲檔案。

頁長度

ID ATTR_PAGELEN

類型 浮點

說明 頁的長度。在測量方法屬性中指定測量單位。

檔案庫名稱

ID ATTR_LIBRARY

類型 字串

說明 檔案庫的名稱。

每吋行數

ID ATTR_LPI

類型 浮點

說明 排存檔中垂直方向每吋的行數。

行距

ID ATTR_LINESPACING

類型 字串

說明 列印時檔案的行式資料記錄間隔的方式。僅針對 *LINE 及 *AFPDSLINE 印表機裝置類型檔案傳回資訊。有效值爲 *SINGLE、*DOUBLE、*TRIPLE 或 *CTLCHAR。

製造商類型及模型

ID ATTR_MFGTYPE

類型 字串

說明 從 SCS 轉換到 ASCII 時指定製造商、類型及模型。

每一用戶端清單的最大工作數

ID ATTR_MAX_JOBS_PER_CLIENT

類型 整數

說明 由用戶端所提供，指出印表機佇列大小的上限。

最大排存輸出記錄

ID ATTR_MAXRECORDS

類型 整數

說明 開啓檔案的同時此檔案可接受的記錄數上限。*NOMAX 的值已編碼爲 0。

測量方法

ID ATTR_MEASMETHOD

類型 字串

說明 用於頁長度及頁寬度屬性的測量方法。有效值爲 *ROWCOL 及 *UOM。

訊息說明

ID ATTR_MSGHELP

類型 char(*)

說明 訊息說明有時候是第二層文字，可使用 retrieve message 要求將它傳回。 系統限制長度爲 3000 個字元 (英文版可供轉換的部分必須小於 30%)。

訊息 ID

ID ATTR_MESSAGEID

類型 字串

說明 訊息 ID。

訊息佇列

ID ATTR_MESSAGE_QUEUE

類型 字串

說明 使用者用於作業訊息之訊息佇列的「整合檔案系統」路徑。「整合檔案系統」路徑的格式爲「/QSYS.LIB/library.LIB/messageque.MSGQ」，其中 *library* 是包含訊息佇列的檔案庫，而 *messageque* 則是訊息佇列的名稱。

訊息回答

ID ATTR_MSGREPLY

類型 字串

說明 訊息回覆。由用戶端所提供的字串，用來回答類型爲 inquiry 的訊息。在擷取訊息的情況下，屬性值由伺服器傳回，且包含用戶端可使用的預設回答。系統限制長度爲 132 個字元。因爲是可變長度的字串，必須以 NULL 作爲結尾。

訊息文字

ID ATTR_MSGTEXT

類型 字串

說明 訊息文字有時候是最上層文字，可使用 retrieve message 要求將它傳回。 系統限制長度爲 132 個字元。

訊息類型

ID ATTR_MSGTYPE

類型 字串

說明 訊息類型是一個 2 位數的 EBCDIC 編碼。有兩種訊息類型可指出是否可「回答擷取的訊息」：04 參考訊息傳送資訊時不要求回答 (而可能要求更正動作)，05 查詢訊息傳送資訊時則要求回覆。

訊息嚴重性

ID ATTR_MSGSEV

類型 整數

說明 訊息嚴重性。爲 00 到 99 的值。值愈高，表示愈嚴重或愈重要的狀況。

多重項目回覆功能

ID ATTR_MULTI_ITEM_REPLY

類型 字串

說明 當用戶端將此屬性值設定爲 *YES 時，可大大地改進列出排存檔作業的效能。預設值爲 *NO。

網路 ID

ID ATTR_NETWORK

類型 字串

說明 建立檔案之系統的網路 ID。

排存檔中的位元組數

ID ATTR_NUMBYTES_SPLF

類型 整數

說明 串流或排存檔中可用的總位元組數。該值指出開始轉換資料前的位元組數。爲了容納檔案大小大於 2**31 - 1 位元組的檔案，這個值是可縮放的；使用者需要將該值乘以 10K，以得到實際的位元組數。此屬性對於以 page-at-a-time 模式檢視的排存檔無效。

讀取/寫入位元組數

ID ATTR_NUMBYTES

類型 整數

說明 讀取作業所讀取的位元組數，或是寫入作業所寫入的位元組數。物件動作決定解譯此屬性的方法。

檔案數

ID ATTR_NUMFILES

類型 整數

說明 輸出佇列上有的排存檔數。

佇列上已啓動寫出器的數量

ID ATTR_NUMWRITERS

類型 整數

說明 輸出佇列上已啓動的寫出器數量。

物件延伸屬性

ID ATTR_OBJEXTATTR

類型 字串

說明 字型資源這類的物件所使用的「延伸」屬性。這個值是透過伺服器上的 WRKOBJ 及 DSPOBJD 指令顯示。伺服器螢幕上的標題可能只指出「屬性」。例如，在字型資源的物件類型中，共用值爲 CDEPAG、CDEFNT 及 FNTCHRSET。

工作中佇列狀態

ID ATTR_ONJOBQSTS

類型 字串

說明 寫出器是否在工作佇列上，因而目前並非正在執行。可能的值爲 *YES 及 *NO。

開啓時間指令

ID ATTR_OPENCMDS

類型 字串

說明 指定使用者是否要將 SCS 開啓時間指令插入排存檔資料之前的資料串流中。有效值爲 *YES 及 *NO。

操作員已控制

ID ATTR_OPCNTRL

類型 字串

說明 是否可讓具有工作控制權限的使用者管理或控制此佇列上的排存檔。有效值爲 *YES 及 *NO。

佇列上的檔案次序

ID ATTR_ORDER

類型 字串

說明 此輸出佇列上排存檔的次序。有效值爲 *FIFO 及 *JOBNBR。

輸出紙匣

ID ATTR_OUTPUTBIN

類型 整數

說明 印表機用來列印輸出的輸出紙匣。其值爲 1 到 65535。*DEVD 的值已編碼爲 0。

輸出優先順序

ID ATTR_OUTPTY

類型 字串

說明 排存檔的優先順序。優先順序的範圍是從 1 (最高) 到 9 (最低)。 有效值爲 0-9，其中 0 代表 *JOB。

輸出佇列

ID ATTR_OUTPUT_QUEUE

類型 字串

說明 輸出佇列的「整合檔案系統」路徑。「整合檔案系統」路徑的格式爲「/QSYS.LIB/library.LIB/queue.QUTQ」，其中 *library* 是包含輸出佇列的檔案庫，而 *queue* 則是輸出佇列的名稱。

輸出佇列狀態

ID ATTR_OUTQSTS

類型 字串

說明 輸出佇列的狀態。有效值爲 RELEASED 及 HELD。

整體狀態

ID ATTR_OVERALLSTS

類型 整數

說明 「邏輯印表機」的整體狀態。「邏輯印表機」提及印表機裝置、輸出佇列及寫出器工作。有效值爲 1 (無法使用)、2 (關閉電源或尚無法使用)、3 (已停止)、4 (等待訊息)、5 (已保留)、6 (停止擱置)、7 (保留擱置)、8 (等待印表機)、9 (等待啓動)、10 (列印中)、11 (等待輸出佇列)、12 (連線擱置)、13 (關閉電源)、14 (無法使用)、15 (處理中) 及 999 (不明)。

溢位行號

ID ATTR_OVERFLOW

類型 整數

說明 於列印在下一頁的溢位資料前所列印的最後一行。

一次一頁

ID ATTR_PAGE_AT_A_TIME

類型 字串

說明 指定是否以 page-at-a-time 模式開啓排存檔。 有效值爲 *YES 及 *NO。

預估頁數

ID ATTR_PAGES_EST

類型 字串

說明 指定預估而非實際的頁數。有效值爲 *YES 及 *NO。

頁面定義

ID ATTR_PAGE_DEFINITION

類型 字串

說明 頁面定義的「整合檔案系統」路徑或特殊值。如果「整合檔案系統」路徑指定的格式爲 /QSYS.LIB/library.LIB/pagedef.PAGDFN，其中 *library* 是頁面定義的檔案庫，而 *pagedef* 則是頁面定義的名稱。有效的特殊值爲 *NONE。

頁次

ID ATTR_PAGENUMBER

類型 整數

說明 從以 page-at-a-time 模式開啓的排存檔所讀取的頁數。

每邊的頁數

ID ATTR_MULTIUP

類型 整數

說明 列印檔案時，列印於每一實體頁上的每一邊的邏輯頁數。有效值爲 1、2 及 4。

紙張來源 1

ID ATTR_PAPER_SOURCE_1

類型 字串

說明 紙張來源 1 中的紙張大小。若未指定此欄位或爲無效的值，則使用特殊值 *MFRTYPMDL。有效值爲 *NONE - 沒有紙張來源 1 或印表機是以手動送紙、*MFRTYPMDL - 使用製造商類型及模型建議的紙張大小、*LETTER (8.5 x 11.0 吋)、*LEGAL (8.5 x 14.0 吋)、*EXECUTIVE (7.25 x 10.5 吋)、*LEDGER (17.0 x 11.0 吋)、*A3 (297mm x 420mm)、*A4 (210mm x 297mm)、*A5 (148mm x 210mm)、*B4 (257mm x 364mm)、*B5 (182mm x 257mm)、*CONT80 (8.0 吋寬的連續報表) 及 *CONT132 (13.2 吋寬的連續報表)。

紙張來源 2

ID ATTR_PAPER_SOURCE_2

類型 字串

說明 紙張來源 2 中的紙張大小。若未指定此欄位或爲無效的值，則使用特殊值 *MFRTYPMDL。有效值爲 *NONE - 沒有紙張來源 2 或印表機是以手動送紙、*MFRTYPMDL - 使用製造商類型及模型建議的紙張大小、*LETTER (8.5 x 11.0 吋)、*LEGAL (8.5 x 14.0 吋)、*EXECUTIVE (7.25 x 10.5 吋)、

*LEDGER (17.0 x 11.0 吋)、*A3 (297mm x 420mm)、*A4 (210mm x 297mm)、*A5 (148mm x 210mm)、*B4 (257mm x 364mm)、*B5 (182mm x 257mm)、*CONT80 (8.0 吋寬的連續報表) 及 *CONT132 (13.2 吋寬的連續報表)。

圖素密度

ID ATTR_PELDENSITY

類型 字串

說明 僅限於字型資源，這個值是指圖素的編碼 (″1″ 代表 240 圖素大小、″2″ 代表 320 圖素大小)。伺服器定義的其它值也具有意義。

字體大小

ID ATTR_POINTSIZE

類型 浮點

說明 此排存檔所列印文字的字體大小。特殊值 *NONE 將編碼爲 0。

列印清晰度

ID ATTR_FIDELITY

類型 字串

說明 列印時執行的一種錯誤處理常式。有效值爲 *ABSOLUTE 及 *CONTENT。

雙面列印

ID ATTR_DUPLEX

類型 字串

說明 列印資訊的方法。有效值爲 *FORMDF、*NO、*YES 及 *TUMBLE。

列印品質

ID ATTR_PRTQUALITY

類型 字串

說明 列印此排存檔時使用的列印品質。有效值爲 *STD、*DRAFT、*NLQ 及 *FASTDRAFT。

列印順序

ID ATTR_PRTSEQUENCE

類型 字串

說明 列印順序。有效值爲 *NEXT。

列印文字

ID ATTR_PRTTEXT

類型 字串

說明 列印於列印輸出每一頁底端及分隔頁上的文字。有效的特殊值包括 *BLANK 及 *JOB。

印表機

ID ATTR_PRINTER

類型 字串

說明 印表機裝置的名稱。

已指定印表機

ID ATTR_PRTASSIGNED

類型 字串

說明 指出是否已指定印表機。有效值爲 1 (指定給特定的印表機)、2 (指定給多重印表機) 及 3 (未指定)。

印表機裝置類型

ID ATTR_PRTDEVTYPE

類型 字串

說明 印表機資料串流類型。有效值爲 *SCS、*IPDS、*USERASCII、*AFPDS 及 *LINE。

印表機檔案

ID ATTR_PRINTER_FILE

類型 字串

說明 印表機檔案的「整合檔案系統」路徑。「整合檔案系統」路徑的格式爲「/QSYS.LIB/library.LIB/printerfile.FILE」，其中 *library* 是 包含印表機檔案的檔案庫，而 *printerfile* 則是印表機檔案的名稱。

印表機佇列

ID ATTR_RMTPRTQ

類型 字串

說明 透過 SNDTCPSPLF (LPR) 傳送排存檔時，目的地印表機佇列的名稱。

支援的發佈資訊顏色

ID ATTR_PUBINF_COLOR_SUP

類型 字串

說明 指出針對此發佈清單項目所支援的顏色。

每分鐘發佈資訊的頁數 (彩色)

ID ATTR_PUBINF_PPM_COLOR

類型 整數

說明 爲此發佈清單項目以彩色模式支援的每分鐘頁數。

每分鐘發佈資訊的頁數 (單色)

ID ATTR_PUBINF_PPM

類型 整數

說明 爲此發佈清單項目以單色模式支援的每分鐘頁數。

發佈資訊雙工支援

ID ATTR_PUBINF_DUPLEX_SUP

類型 字串

說明 此發佈清單項目的雙工支援指示器。

發佈資訊位置

ID ATTR_PUBINF_LOCATION

類型 字串

說明 此發佈清單項目的位置說明。

遠端位置名稱

ID ATTR_RMTLOCNAME

類型 字串

說明 印表機裝置位置名稱。

記錄長度

ID ATTR_RECLENGTH

類型 整數

說明 記錄長度。

減少輸出

ID ATTR_REDUCE

類型 字串

說明 多重邏輯頁在實體頁的每一端列印的方式。有效值爲 *TEXT 或 ????。

遠端系統

ID ATTR_RMTSYSTEM

類型 字串

說明 遠端系統名稱。有效的特殊值有 *INTNETADR。

置換不可列印的字元

ID ATTR_RPLUNPRT

類型 字串

說明 是否將無法列印的字元置換爲其它的字元。有效值爲 *YES 或 *NO。

置換字元

ID ATTR_RPLCHAR

類型 字串

說明 置換任何不可列印字元的字元。

重新啓動列印

ID ATTR_RESTART

類型 整數

說明 重新啓動列印。有效值爲 -1、-2、-3 或要重新啓動的頁次。*STRPAGE 的值已編碼爲 -1、*ENDPAGE 的值已編碼爲 -2 及 *NEXT 的值已編碼爲 -3。

馬鞍型裝訂縫針數

ID ATTR_SADDLESTITCH_NUMSTAPLES

類型 整數

說明 沿著完成的作業軸所套用的裝訂針數。

馬鞍型縫針參照

ID ATTR_SADDLESTITCH_REF

類型 字串

說明 一或多個裝訂沿著完成作業軸導向媒體，該軸是定位在與參照邊平行的媒體中央。有效值爲 *NONE、*DEVD、*TOP 及 *LEFT。

儲存排存檔

ID ATTR_SAVESPLF

類型 字串

說明 在寫入排存檔後是否要儲存。有效值爲 *YES 及 *NO。

探查位移

ID ATTR_SEEKOFF

類型 整數

說明 探查位移。接受與探查原點相關的正負值。

探查原點

ID ATTR_SEEKORG

類型 整數

說明 有效值有 1 (開頭或頂端)、2 (目前) 及 3 (結尾及底端)。

傳送優先順序

ID ATTR_SENDPTY

類型 字串

說明 傳送優先順序。有效值爲 *NORMAL 及 *HIGH。

分隔頁

ID ATTR_SEPPAGE

類型 字串

說明 接受使用者是否列印標籤頁的選項。有效值爲 *YES 或 *NO。

來源匣

ID ATTR_SRCDRWR

類型 整數

說明 選取自動裁剪送紙時使用的紙匣。有效值爲 -1、-2 及 1-255。*E1 的值已編碼爲 -1，以及 *FORMDF 的值已編碼爲 -2。

排存 SCS

ID ATTR_SPLSCS

類型 Long

說明 判定建立排存檔期間如何使用 SCS 資料。

排存資料

ID ATTR_SPOOL

類型 字串

說明 是否排存印表機裝置的輸出資料。有效值爲 *YES 及 *NO。

排存檔建立鑑別方法

ID ATTR_SPLF_AUTH_METHOD

類型 整數

說明 指出用來建立此排存檔的用戶端鑑別方法。有效值包括 x'00'(*NONE)、x'01'(*REQUESTER)、x'02'(*BASIC)、x'03'(*CERTIFICATE) 及 'x'04'(*DIGEST)。

排存檔建立安全性方法

ID ATTR_SPLF_SECURITY_METHOD

類型 字串

說明 指出用來建立此排存檔的安全性方法。有效值爲 x'00'(*NONE)、x'01'(*SSL3) 及 x'02'(*TLS)。

排存檔名稱

ID ATTR_SPOOLFILE

類型 字串

說明 排存檔的名稱。

排存檔號碼

ID ATTR_SPLFNUM

類型 整數

說明 排存檔的編號。允許的特殊值爲 -1 和 0。*LAST 值已編碼爲 -1，*ONLY 值已編碼爲 0。

排存檔狀態

ID ATTR_SPLFSTATUS

類型 字串

說明 排存檔的狀態。有效值爲 *CLOSED、*HELD、*MESSAGE、*OPEN、*PENDING、*PRINTER、*READY、*SAVED 及 *WRITING。

排存輸出時程表

ID ATTR_SCHEDULE

類型 字串

說明 僅限於排存檔，指定寫出器可使用排存檔的時機。有效值爲 *IMMED、*FILEEND 及 *JOBEND。

由使用者啓動

ID ATTR_STARTEDBY

類型 字串

說明 啓動寫出器之使用者的名稱。

起始頁

ID ATTR_STARTPAGE

類型 整數

說明 開始列印排存檔的頁次。有效值爲 -1、0、1 或頁次。*ENDPAGE 的值已編碼爲 -1。0 值表示由第 1 頁開始列印。1 值則表示整個檔案列印。

系統建立位置

ID ATTR_SYSTEM

類型 字串

說明 建立排存檔所在的系統名稱。無法判定排存檔建立之處的系統名稱時，則使用接收系統名稱。

本文說明

ID ATTR_DESCRIPTION

類型 字串

說明 說明 AS400 物件之案例的說明文字。

開啓檔案時間

ID ATTR_TIMEOPEN

類型 字串

說明 針對排存檔，則這是開啓此排存檔的時間。針對 AFP 資源，則是前次修改物件的時間。時間是以 HH MM SS 的格式所編碼的字串。

排存檔工作建立結束時間

ID ATTR_TIME_END

類型 字串

說明 系統結束時排存檔建立工作的時間。當欄位「啓動排存檔建立日期」使用特殊值 *ALL 或當欄位「結束排存檔建立日期」使用特殊值 *LAST 時，必須將此欄位設定爲空白。若在欄位「結束排存檔建立日期」指定日期，則必須設定此欄位的值。時間必須爲 HHMMSS 格式，定義如下：

- HH - 小時
- MM - 分鐘
- SS - 秒

寫出器開始處理排存檔時間

ID ATTR_TIME_WTR_BEGAN_FILE

類型 字串

說明 指定寫出器開始處理排存檔的時間。時間是以 HH MM SS 的格式所編碼的字串。

寫出器完成處理排存檔時間

ID ATTR_TIME_WTR_CMPL_FILE

類型 字串

說明 指定寫出器完成處理排存檔的時間。時間是以 HH MM SS 的格式所編碼的字串。

總頁數

ID ATTR_PAGES

類型 整數

說明 排存檔所含的頁數。

轉換 SCS 為 ASCII

ID ATTR_SCS2ASCII

類型 字串

說明 是否將列印資料從 SCS 轉換爲 ASCII。有效值爲 *YES 及 *NO。

測量單位

ID ATTR_UNITOFMEAS

類型 字串

說明 用於指定距離的測量單位。有效值爲 *CM 及 *INCH。

使用者註解

ID ATTR_USERCMT

類型 字串

說明 長度爲 100 個字元的使用者指定註解，用來說明排存檔。

使用者資料

ID ATTR_USERDATA

類型 字串

說明 長度爲 10 個字元的使用者指定資料，用來說明排存檔。有效的特殊值包括 *SOURCE。

使用者定義資料

ID ATTR_USRDFNDTA

類型 字串

說明 由使用者應用程式或處理排存檔之使用者指定程式所利用的使用者定義資料。可接受所有的字元。最大之大小爲 255。

使用者定義檔案

ID ATTR_USRDEFFILE

類型 字串

說明 建立的排存檔是否使用 API。有效值爲 *YES 或 *NO。

使用者定義物件

ID ATTR_USER_DEFINED_OBJECT

類型 字串

說明 處理排存檔之使用者應用程式所利用的使用者定義物件的「整合檔案系統」路徑。如果「整合檔案系統」路徑的格式爲「/QSYS.LIB/library.LIB/object.type」，其中 *library* 是包含物件或 %LIBL%。*object* 爲物件的名稱，而 *type* 則是物件類型。*type* 的有效值包括：DTAARA、DTAQ、FILE、PSFCFG、USRIDX、USRQ 及 USRSPC。*NONE 字串表示未使用使用者定義的物件。

使用者定義選項

ID ATTR_USEDFNOPTS

類型 字串

說明 由處理排存檔之使用者應用程式所利用的使用者定義選項。最多可指定 4 個選項，每一個值的長度爲 char(10)。可接受所有的字元。

使用者驅動程式資料

ID ATTR_USRDRVPGMDTA

類型 字串

說明 與使用者驅動程式一起使用的使用者資料。可接受所有的字元。最大爲 5000 個字元。

使用者驅動程式

ID ATTR_USER_DRIVER_PROG

類型 字串

說明 處理排存檔之使用者定義驅動程式程式所利用的「整合檔案系統」路徑。「整合檔案系統」路徑的格式爲「/QSYS.LIB/library.LIB/program.PGM」，其中 *library* 爲包含程式的檔案庫名稱，而 *program* 則爲程式名稱。*library* 可能爲特殊值 %LIBL% 與 %CURLIB%，或者是特定的程式庫名稱。*NONE 字串用來表示尚未定義驅動程式。

使用者 ID

ID ATTR_TOUSERID

類型 字串

說明 接收排存檔之使用者的使用者 ID。

使用者 ID 位址

ID ATTR_TOADDRESS

類型 字串

說明 接收排存檔之使用者的使用者位址。

使用者轉換程式

ID ATTR_USER_TRANSFORM_PROG

類型 字串

說明 驅動程式處理排存檔資料之前，轉換該資料的使用者定義轉換程式之「整合檔案系統」路徑。「整合檔案系統」路徑的格式爲「/QSYS.LIB/library.LIB/program.PGM」，其中 *library* 爲包含程式的檔案庫名稱，而 *program* 則爲程式名稱。*library* 可能爲特殊值 %LIBL% 與 %CURLIB%，或者是特定的程式庫名稱。*NONE 字串用來表示尚未定義轉換程式。

檢視清晰度

ID ATTR_VIEWING_FIDELITY

類型 字串

說明 檢視排存檔資料頁時發生的處理程序 (資料爲 page-at-a-time 模式)。 有效值爲 *ABSOLUTE 及 *CONTENT (預設)。如果要於處理現行頁之前先處理所有的非掃瞄資料 (指令)，請使用 *AUSOLUTE。針對 SCS 檔案， *CONTENT 僅用來處理開啓時間指令及現行頁。針對 AFPDS 檔案， *CONTENT 用來處理資料的首頁及現行頁。

VM/MVS 類別

ID ATTR_VMMVSCLASS

類型 字串

說明 VM/MVS 類別。有效值爲 A-Z 及 0-9。

等待資料狀態

ID ATTR_WTNGDATASTS

類型 字串

說明 寫出器是否已寫入排存檔中目前的所有資料，並且等待更多資料。可能的值爲 *NO - 寫出器未等待更多資料，*YES - 寫出器已寫入排存檔中目前的所有資料，並且等待更多資料。當寫出器以指定的 SCHEDULE(*IMMED) 產生開啓排存檔時，會發生此情況。

等待裝置狀態

ID ATTR_WTNGDEVSTS

類型 字串

說明 寫出器是否正等待從直接列印到印表機之工作取得裝置。可能的值爲 *NO - 寫出器並未等待裝置，*YES - 寫出器正等待裝置。

等待訊息狀態

ID ATTR_WTNGMSGSTS

類型 字串

說明 寫出器正等待對查詢訊息的回答。值爲 *NO 及 *YES。

自動結束寫出器時間

ID ATTR_WTRAUTOEND

類型 字串

說明 要自動結束寫出器的時間。有效值爲 *NORDYF 及 *FILEEND。屬性「自動結束寫出器」必須設定爲 *YES。

結束寫出器時間

ID ATTR_WTREND

類型 字串

說明 何時要結束寫出器。有效值爲 *CNTRLD、*IMMED 及 *PAGEEND。這與何時自動結束寫出器不同。

保留檔案時間

ID ATTR_HOLDTYPE

類型 字串

說明 何時要保留排存檔。有效值爲 *IMMED 及 *PAGEEND。

頁寬度

ID ATTR_PAGEWIDTH

類型 浮點

說明 頁的寬度。在測量方法屬性中指定測量單位。

工作站自訂物件

ID ATTR_WORKSTATION_CUST_OBJECT

類型 字串

說明 工作站自訂物件的「整合檔案系統」路徑。「整合檔案系統」路徑的格式爲「/QSYS.LIB/library.LIB/custobj.WSCST」，其中 *library* 是包含工作站自訂物件的檔案庫， 而 *custobj* 則是工作站自訂物件的名稱。

寫出器工作名稱

ID ATTR_WRITER

類型 字串

說明 寫出器工作的名稱。

寫出器工作編號

ID ATTR_WTRJOBNUM

類型 字串

說明 寫出器工作的編號。

寫出器工作狀態

ID ATTR_WTRJOBSTS

類型 字串

說明 寫出器工作的狀態。有效值爲 STR、END、JOBQ、HLD 及 MSGW。

寫出器工作使用者名稱

ID ATTR_WTRJOBUSER

類型 字串

說明 啓動寫出器工作之使用者的名稱。

寫出器已啓動

ID ATTR_WTRSTRTD

類型 字串

說明 指出是否爲此印表機啓動了寫出器。值 1 - 已啓動寫出器，0 - 未啓動寫出器。

寫出器起始頁

ID ATTR_WTRSTRPAGE

類型 整數

說明 當寫出器工作啓動時，指定要從第一個排存檔列印之首頁的頁次。這僅只於寫出器啓動時同時指定排存檔名稱時有效。

寫入狀態

ID ATTR_WRTNGSTS

類型 字串

說明 指出列印寫出器是否於寫入狀態中。值爲 *YES - 寫出器在寫入狀態，*NO - 寫出器不在寫入狀態，*FILE - 寫出器正寫入檔案分隔字元。

網路列印伺服器物件屬性

NPS CCSID

ID ATTR_NPSCCSID

類型 整數

說明 「網路列印伺服器」預期用來編碼所有字串的 CCSID。

NPS 層次

ID ATTR_NPSLEVEL

說明 「網路列印伺服器」的版本、版次及修正層次。屬性是編碼爲 VXRYMY 的字串 (例如，V3R1M0)，其中

```
X 為 (0..9)
 Y 為 (0..9,A..Z)
```

複製排存檔:

您可以使用 SpooledFile 類別的複製方法，來建立 SpooledFile 物件所代表之排存檔的複本。使用 SpooledFile.copy() 可執行下列動作：

- 在原始排存檔相同的系統中以及相同的輸出佇列上建立新的排存檔
- 傳回參照至新的排存檔

SpooledFile.copy() 是一個新的方法，只有在您下載了 JTOpen 3.2 或更新版本，或者是套用了 i5/OS 修訂程式之後，才可使用。建議您最好下載並使用 JTOpen。如需詳細資訊，請參閱下列資訊：

IBM Toolbox for Java and JTOpen: Downloads 

IBM Toolbox for Java and JTOpen: Service Packs 

複製方法會在網路列印伺服器工作中使用「建立排存檔 (QSPCRTSP) API」，以建立排存檔精確無誤的抄本。您只需要唯一的建立日期與時間，以保留排存檔新建複本的識別資訊。如需 QSPCRTSP API 的詳細資訊，請參閱下列資訊：

建立排存檔 (QSPCRTSP) API

若將輸出佇列指定爲複製方法的參數，即可在指定輸出佇列的第一個位置上建立排存檔的複本。輸出佇列與原始排存檔必須位於相同的系統

範例：使用 SpooledFile.copy() 來複製排存檔

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

本範例將說明如何使用 SpooledFile.copy()，將排存檔複製到您要複製的檔案所在的相同佇列上。當您要將最新複製的排存檔遞送到特定的輸出佇列時，請將輸出佇列當作參數傳遞到複製方法中：

```
SpooledFile newSplf = new sourceSpooledFile.copy(<outqname>);
```

其中 <outqname> 爲 OutputQueue 物件。

```
public static void main(String args[]) {
   // Create the system object
   AS400 as400 = new AS400(<systemname>,<username>, <password>);
   // Identify the output queue that contains the spooled file you want to copy.
   OutputQueue outputQueue =
     new OutputQueue(as400, "/QSYS.LIB/QUSRSYS.LIB/<outqname>.OUTQ");

   // Create an array that contains all the elements required to
   // uniquely identify a spooled file on the iSeries server.
   String[][] splfTags = { {
      <spoolfilename>,
      <spoolfilenum>,
      <jobname>,
      <username>,
      <jobnumber>,
      // Note that <systemname>,<date>, and <time> are optional.
      // If you do not include them, remove the corresponding
      // splfTags[i],[j], where j has the value of 5,6, or 7.
      <systemname>,
      <date>,
      <time>},
   };

   // Print the information that identifies the spooled file to System.out
   for ( int i=0; i<splfTags.length; i++) {
      System.out.println("Copying -> " + splfTags[i][0] + ","
                                       + splfTags[i][1] + ","
                                       + splfTags[i][2] + ","
                                       + splfTags[i][3] + ","
                                       + splfTags[i][4] + ","
                                       + splfTags[i][5] + ","
                                       + splfTags[i][6] + ","
                                       + splfTags[i][7] );

   // Create the SpooledFile object for the source spooled file.
      SpooledFile sourceSpooledFile =
         new SpooledFile(as400,
                         splfTags[i][0],
                         Integer.parseInt(splfTags[i][1]),
                         splfTags[i][2],
                         splfTags[i][3],
```

```
                        splfTags[i][5],
                        splfTags[i][6],
                        splfTags[i][7] );
    }

    // Copy the spooled file, which creates a new SpooledFile object.
    // To route the copy of the spooled file to a specific output queue,
    // use the following code:
    // SpooledFile newSplf = new sourceSpooledFile.copy(<outqname>);
    // where <outqname> is an OutputQueue object. Specify the output
    // queue in the following way:
    // OutputQueue outputQueue =
    //    new OutputQueue(as400, "/QSYS.LIB/QUSRSYS.LIB/<outqname>.OUTQ");
    try { SpooledFile newSplf = new sourceSpooledFile.copy();
    }

    catch (Exception e) {
    }
```

Javadoc 參考文件

如需有關 SpooledFile.copy() 的詳細資訊，請參閱下列 Javadoc 參考文件：

SpooledFile copy() 方法

建立新的排存檔:

您可以使用 SpooledFileOutputStream 類別來建立新的伺服器排存檔。從標準 JDK java.io.OutputStream 類別中衍生此類別；建構此類別之後，可在有使用 OutputStream 的任何地方使用此類別。

建立新 SpooledFileOutputStream 時，呼叫程式可能指定下列項目：

- 要使用的印表機檔案
- 要存放排存檔的輸出佇列
- PrintParameterList 物件，可包含參數來置換印表機檔案中的欄位

這些參數全部都是選用的 (呼叫程式可能對任何一個或全部參數傳送空值)。如果未指定印表機檔案，則網路列印伺服器會使用預設網路列印印表機檔案 QPNPSPRTF。此處使用輸出佇列參數是爲了方便起見；也可在 PrintParameterList 中指定它。如果這兩處都指定輸出佇列參數，則 PrintParameterList 欄位會置換輸出佇列參數。請參閱 SpooledFileOutputStream 建構子的文件，取得完整的清單，當中有列出哪些屬性可設定。

請使用一個 write() 方法將資料寫入排存檔。SpooledFileOutputStream 物件會緩衝資料，然後等到結束輸出串流或緩衝區已滿之後再傳送資料。基於兩個理由需要執行緩衝：

- 容許自動資料鍵入 (請參閱排存檔中的資料串流類型) 來分析滿緩衝區資料，以判斷資料類型。
- 因爲不是每一個寫入要求都傳送至伺服器，所以可使輸出串流更快輸出。

請使用 flush() 方法以強制資料寫入伺服器。

呼叫程式將資料寫入到新的排存檔之後，會呼叫 close() 方法來關閉該排存檔。排存檔關閉之後，便無法對此檔案寫入資料。一旦關閉排存檔後，藉由立刻呼叫 getSpooledFile() 方法，呼叫程式即可取得代表排存檔的 SpooledFile 物件的參照。

排存檔中的資料串流類型

使用排存檔的「印表機資料類型」屬性來設定要存放在排存檔的資料類型。如果呼叫程式未指定印表機資料類型，則預設值會使用自動資料鍵入。本方法會查看排存檔資料中最前面的數千個位元組，來判斷它是適合

「SNA 字元串流 (SCS)」還是「進階功能列印資料串流 (AFPDS)」資料串流架構，然後適當地設定屬性。如果排存檔的位元組不符合這些架構，則資料會標示成 *USERASCII。大部分時間都可使用自動資料鍵入。除非呼叫程式發生無法使用自動資料鍵入的特殊情況，否則，呼叫程式通常會使用自動資料鍵入。在那些情況中，呼叫程式可設定「印表機資料類型」屬性爲特定值 (例如，*SCS)。如果呼叫程式要使用印表機檔案中的印表機資料，則呼叫程式必須使用特殊值 *PRTF。 建立排存檔時如果呼叫程式置換預設資料類型，請確定存放於排存檔的資料符合此資料類型屬性。若將非 SCS 資料存放於某個標示爲接收 SCS 資料的排存檔，則會從主電腦發出錯誤訊息並失去排存檔。

一般而言，本屬性可擁有三個值：

- ***SCS** - EBCDIC，以文字爲主的印表機資料串流。
- ***AFPDS** (進階功能呈現™資料串流) - 伺服器支援的另一種資料串流。*AFPDS 可包含文字、影像以及圖形，而且可在頁面區段中使用外部資源 (如頁面套印格式) 和外部影像。
- ***USERASCII** - 指任何非 SCS 和非 AFPDS 印表機資料，伺服器只負責傳送。Postscript 和 HP-PCL 資料串流是存放在 *USERASCII 排存檔的範例資料串流。

範例

下列範例說明使用排存檔的各種方式。第一個範例爲您說明，如何從輸入串流中建立伺服器的排存檔。第二個範例爲您說明，如何使用 SCS3812Writer 類別產生 SCS 資料串流，以及如何將串流寫入伺服器的排存檔中。

第 462 頁的『範例：建立排存檔』

第 463 頁的『範例：建立 SCS 排存檔』

產生 SCS 資料串流:

若要產生在連接至伺服器之印表機上列印的排存檔，則必須建立「SNA 字元串流 (SCS)」資料串流。(SCS 是文字型的 EBCDIC 資料串流，可在 SCS 印表機、IPDS™ 印表機或 PC 印表機上列印。)在伺服器中使用模擬程式或主電腦列印轉換來轉換之後可列印出 SCS。

您可以使用 SCS 寫出器類別來產生這樣的 SCS 資料串流。SCS 寫出器類別會將 Java Unicode 字元及格式化選項轉換爲 SCS 資料串流。五個 SCS 寫出器類別產生不同層次的 SCS 資料串流。呼叫程式會根據呼叫程式或一般使用者要列印的最終目的地印表機，選擇相符的寫出器。

使用下列 SCS 寫出器類別來產生 SCS 印表機資料串流：

SCS 寫出器類別	說明
SCS5256Writer	最簡單的 SCS 寫出器類別。支援文字、回車、換行、新行、換頁、絕對水平及垂直定位、相對水平及垂直定位以及設定垂直格式。
SCS5224Writer	擴充 5256 寫出器，並新增方法來設定每吋字元數 (CPI) 以及每吋行數 (LPI)。
SCS5219Writer	擴充 5224 寫出器，並新增對下列項目的支援：左邊距、底線、紙張規格 (紙張或信封)、紙張大小、列印品質、字碼頁、字集、來源紙匣號碼以及目的紙匣號碼。
SCS5553Writer	擴充 5219 寫出器，並新增對於字元旋轉、格線以及字型比例的支援。5553 是雙位元組字集 (DBCS) 資料串流。
SCS3812Writer	擴充 5219 寫出器，並新增對粗體、雙面列印、文字方向以及字型的支援。

若要建構 SCS 寫出器，則呼叫程式需要輸出串流和編碼 (可選用的)。資料串流會寫入輸出串流。若要建立 SCS 排存檔，則呼叫程式必須先建構 SpooledFileOutputStream，然後使用 SpooledFileOutputStream 來建構 SCS 寫出器物件。此編碼參數提供目標 EBCDIC 編碼字集識別字 (CCSID) 來轉換字元。

一旦建構寫出器後，請使用 write() 方法來輸出文字。請使用 carriageReturn()、lineFeed() 和 newLine() 等方法將寫入游標定位在頁中。請使用 endPage() 方法來結束現行頁並起始新頁。

寫入所有資料之後，請使用 close() 方法結束資料串流並關閉輸出串流。

範例

下列範例說明如何使用 SCS3812Writer 類別來產生 SCS 資料串流，以及如何將串流寫入伺服器的排存檔中：

範例：建立 SCS 排存檔

讀取排存檔及 AFP 資源:

您可以使用 PrintObjectInputStream 類別從伺服器中讀取排存檔的原始內容或「進階功能列印 (AFP)」資源。此類別擴充標準 JDK java.io.InputStream 類別，因此可在有使用 InputStream 的任何地方使用它。

要取得 PrintObjectInputStream 物件，可呼叫 SpooledFile 類別的實例中的 getInputStream() 方法或呼叫 AFPResource 類別的實例中的 getInputStream() 方法。「版本 3 版次 2 (V3R2)」、V3R7 以及更新版本的 i5/OS 都支援取得排存檔的輸入串流。V3R7 及更新版本都支援取得 AFP 資源的輸入串流。

使用其中一個 read() 方法，從輸入串流中讀取資料。這些方法會傳回實際讀取的位元組數，如果沒有讀取到位元組或到達檔案結尾，則傳回 -1。

您可以使用 PrintObjectInputStream 的 available() 方法來傳回排存檔或 AFP 資源中的總位元組數。PrintObjectInputStream 類別支援標示輸入串流，以便 PrintObjectInputStream 一律以 markSupported() 方法傳回 true。呼叫程式可以使用 mark() 和 reset() 方法，將輸入串流中的現行讀取位置往後移。請使用 skip() 方法在輸入串流中向前移動讀取位置但不讀取資料。

範例

下列範例說明如何使用 PrintObjectInputStream 來讀取現有的伺服器排存檔

範例：讀取排存檔

使用 PrintObjectPageInputStream 與 PrintObjectTransformedInputStream 讀取排存檔:

您可以使用 PrintObjectPageInputStream 類別，逐頁讀取伺服器 AFP 及 SCS 排存檔中的資料。

使用 getPageInputStream() 方法可以取得 PrintObjectPageInputStream 物件。

使用其中一個 read() 方法，從輸入串流中讀取資料。所有這些方法會傳回實際讀取的位元組數，如果沒有讀取到位元組或到達頁結尾，則傳回 -1。

您可以使用 PrintObjectPageInputStream 的 available() 方法來傳回現行頁的總位元組數目。PrintObjectPageInputStream 類別支援標示輸入串流，以便使 PrintObjectPageInputStream 一律從 markSupported() 方法傳回 true。呼叫程式可以使用 mark() 和 reset() 方法將輸入串流中的現行讀取位置向後移，以便後續讀取可以重新讀取同一位元組。呼叫程式可以使用 skip() 方法在輸入串流中向前移動讀取位置但不讀取資料。

然而，當想要轉換整個排存檔資料串流時，請使用 PrintObjectTransformedInputStream 類別。

範例

下列範例將說明，如何使用 PrintObjectPageInputStream 及 PrintObjectTransformedInputStream，在讀取排存檔時獲得不同的轉換：

第 465 頁的『範例：讀取與轉換排存檔』

產品授權

ProductLicense 類別讓您能夠要求 iSeries 上所安裝產品的授權。爲了與其他 iSeries 授權使用者相容，在要求或釋出授權時，類別是透過 iSeries 產品授權支援執行的。

類別並不強制實施授權原則，但會傳回足夠的資訊，使得應用程式能夠強制實施原則。當要求授權時，ProductLicense 類別 會傳回要求的狀態 -- 授權授予或者拒絕。如果要求遭到拒絕，應用程式必須停用需要授權的行爲，因爲 IBM Toolbox for Java 不知道應停用哪個功能。

請使用 ProductLicense 類別與 iSeries 授權支援來強制應用程式的授權：

- 您的應用程式的伺服器端應使用 iSeries 授權支援來註冊您的產品及授權條款。
- 您的應用程式的從屬端使用 ProductLicense 物件來要求和釋出授權。

範例：ProductLicense 實務

例如，假定您的客戶購買了您產品的 15 套並行使用授權。並行使用表示有 15 位使用者可同時使用該產品，但不限定 15 位特定的使用者。可以爲 組織內的任何 15 位使用者。此資訊是以 iSeries 授權支援註冊的。當使用者連接您的應用程式時，會使用 ProductLicense 類別來要求授權。

- 當並行使用者人數少於 15，要求會順利完成，您的應用程式也就執行。
- 當有第 16 位使用者連線時，ProductLicense 要求便告失敗。您的應用程式就會顯示錯誤訊息，並且結束。

當有使用者停止執行應用程式時，您的應用程式就會經由 ProductLicense 類別釋出授權。授權即可供他人使用。

如需更多資訊和程式碼範例，請參閱 ProductLicense javadoc。

ProgramCall 類別

ProgramCall 類別可讓 Java 程式呼叫 iSeries 程式。您可以使用 ProgramParameter 類別指定輸入、輸出和輸入/輸出參數。當執行程式時，輸出及輸入/輸出參數會含有 iSeries 程式所傳回的資料。如果 iSeries 程式無法順利執行，Java 程式可以擷取產生的 iSeries 訊息，作爲 AS400Message 物件清單。

必要的參數如下：

- 要執行的程式與參數
- 代表具有該程式之 iSeries 伺服器的 AS400 物件。

經由 setProgram() 方法，程式名稱和參數清單可以設定在建構子上，或設定在 run() 方法上。run() 會方法呼叫該程式。

若使用 ProgramCall 類別，AS400 物件就會連接到 iSeries 伺服器。有關管理連線的資訊，請參閱管理連線。

範例：使用 ProgramCall

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

下面範例說明如何使用 ProgramCall 類別：

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a program object. I choose
                    // to set the program to run later.
ProgramCall pgm = new ProgramCall(sys);

                    // Set the name of the program.
                    // Because the program does not take
                    // any parameters, pass null for the
                    // ProgramParameter[] argument.
pgm.setProgram(QSYSObjectPathName.toPath("MYLIB", "MYPROG", "PGM"));

                    // Run the program. My program has
                    // no parms. If it fails to run, the failure
                    // is returned as a set of messages
                    // in the message list.
if (pgm.run() != true)
{
                    // If you get here, the program
                    // failed to run. Get the list of
                    // messages to determine why the
                    // program didn't run.
   AS400Message[] messageList = pgm.getMessageList();

                    // ... Process the message list.
}

                    // Disconnect since I am done
                    // running programs
sys.disconnectService(AS400.COMMAND);
```

ProgramCall 物件需要程式的整合檔案系統路徑名稱。

預設行爲是即使當 Java 程式與 iSeries 程式位於同一部伺服器上，也可以讓 iSeries 程式在不同的伺服器工作中執行。您可以置換預設行爲，並使用 setThreadSafe() 方法，在 Java 工作中執行 iSeries 程式。

使用 ProgramParameter 物件

您可以使用 ProgramParameter 物件，在 Java 程式與 iSeries 程式之間傳遞參數資料。以 setInputData() 方法設定輸入資料。在執行程式後，可以用 getOutputData() 方法擷取輸出資料。每一個參數都是一個位元組陣列。Java 程式必須在 Java 與 iSeries 格式之間轉換位元組陣列。資料轉換類別可提供轉換資料的方法。參數會新增到 ProgramCall 物件作爲清單。

範例：使用 ProgramParameter

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

下列範例說明如何使用 ProgramParameter 物件來傳送參數資料。

```
                    // Create an AS400 object
AS400 sys = new AS400("mySystem.myCompany.com");

                    // My program has two parameters.
                    // Create a list to hold these
                    // parameters.
ProgramParameter[] parmList = new ProgramParameter[2];

                    // First parameter is an input
                    // parameter
byte[] key = {1, 2, 3};
parmList[0] = new ProgramParameter(key);
```

```
                    // Second parameter is an output
                    // parameter. A four-byte number
                    // is returned.
parmList[1] = new ProgramParameter(4);

                    // Create a program object
                    // specifying the name of the
                    // program and the parameter list.
ProgramCall pgm = new ProgramCall(sys, "/QSYS.LIB/MYLIB.LIB/MYPROG.PGM", parmList);

                    // Run the program.
if (pgm.run() != true)
{

                    // If the iSeries cannot run the
                    // program, look at the message list
                    // to find out why it didn't run.
   AS400Message[] messageList = pgm.getMessageList();

}
else
{
                    // Else the program ran. Process the
                    // second parameter, which contains
                    // the returned data.

                    // Create a converter for this
                    // iSeries data type
   AS400Bin4 bin4Converter = new AS400Bin4();

                    // Convert from iSeries type to Java
                    // object. The number starts at the
                    // beginning of the buffer.
   byte[] data = parmList[1].getOutputData();
   int i = bin4Converter.toInt(data);
}

                    // Disconnect since I am done
                    // running programs
sys.disconnectService(AS400.COMMAND);
```

QSYSObjectPathName 類別

您可以使用 QSYSObjectPathName 類別來代表整合檔案系統中的物件。使用這個類別來建立一個整合檔案系統名稱，或將整合檔案系統名稱剖析爲它的元件。

數個 IBM Toolbox for Java 類別必須要有一個整合檔案系統路徑名稱，才能使用。請使用 QSYSObjectPathName 物件，來建置名稱。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

下面範例說明如何使用 QSYSObjectPathName 類別：

範例 1： ProgramCall 物件需要要呼叫的伺服器程式的整合檔案系統名稱。QSYSObjectPathName 物件是用來建置名稱。若要使用 QSYSObjectPathName，呼叫檔案庫 REPORTS 中的程式 PRINT_IT：

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a program call object.
ProgramCall pgm = new ProgramCall(sys);

                    // Create a path name object that
                    // represents program PRINT_IT in
```

```
                    // library REPORTS.
    QSYSObjectPathName pgmName = new QSYSObjectPathName("REPORTS",
                                                        "PRINT_IT",
                                                        "PGM");

                    // Use the path name object to set
                    // the name on the program call
                    // object.
    pgm.setProgram(pgmName.getPath());

                    // ... run the program, process the
                    // results
```

範例 2： 如果 AS400 物件的名稱只用一次，則 Java 程式可使用 toPath() 方法建置該名稱。這個方法比建立 QSYSObjectPathName 物件更有效。

```
                    // Create an AS400 object.
    AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a program call object.
    ProgramCall pgm = new ProgramCall(sys);

                    // Use the toPath method to create
                    // the name that represents program
                    // PRINT_IT in library REPORTS.
    pgm.setProgram(QSYSObjectPathName.toPath("REPORTS",
                                             "PRINT_IT",
                                             "PGM"));

                    // ... run the program, process the
                    // results
```

範例 3： 在此範例中，提供了一個整合檔案系統路徑給 Java 程式。QSYSObjectPathName 類別可用來將這個名稱剖析爲它的元件：

```
                    // Create a path name object from
                    // the fully qualified integrated
                    // file system name.
    QSYSObjectPathName ifsName = new QSYSObjectPathName(pathName);

                    // Use the path name object to get
                    // the library, name and type of
                    // server object.
    String library = ifsName.getLibraryName();
    String name    = ifsName.getObjectName();
    String type    = ifsName.getObjectType();
```

記錄層次存取

記錄層次存取類別可提供執行下列的能力：

- 指定下列其中一項來建立 iSeries 實體檔案：
  - 記錄長度
  - 現存的資料說明規格 (DDS) 原始檔
  - RecordFormat 物件
- 從 iSeries 實體檔案或邏輯檔案中擷取記錄格式，或從 iSeries 多重格式邏輯檔案中擷取記錄格式。

  註: 不會完整地擷取檔案的記錄格式。 表示當設定 AS400File 物件的記錄格式時，將使用所擷取的記錄格式。僅在擷取足夠資訊時，才能描述檔案記錄的內容。不會擷取記錄格式資訊，如直欄標頭與別名。
- 根據記錄編號或索引來依序存取 iSeries 檔中的記錄。
- 將記錄寫入 iSeries 檔中。

- 根據記錄編號或索引來依序更新 iSeries 檔中的記錄。
- 根據記錄編號或索引來依序刪除 iSeries 檔中的記錄。
- 針對不同存取類型來鎖定 iSeries 檔。
- 使用確定控制，讓 Java 程式執行下列動作：
  - 啓動此連接的確定控制。
  - 針對不同檔案來指定不同確定控制鎖定層次。
  - 確定並回轉異動。
- 刪除 iSeries 檔。
- 從 iSeries 檔刪除成員。

註: 記錄層次存取類別不支援邏輯結合檔或空字元索引欄位。

下列類別執行這些功能：

- AS400File 類別是記錄層次存取類別的抽象基礎類別。它提供循序記錄存取、建立及刪除檔案與成員以及確定控制活動的方法。
- KeyedFile 類別代表按照索引存取的 iSeries 檔。
- SequentialFile 類別代表按照記錄編號存取的 iSeries 檔。
- AS400FileRecordDescription 類別會提供擷取 iSeries 檔之記錄格式的方法。

記錄層次存取類別需要一個 AS400 物件來代表有資料庫檔案的系統。使用記錄層次存取類別可使 AS400 物件連接到 iSeries。有關管理連線的資訊，請參閱管理連線。

記錄層次存取類別需要資料庫檔案的整合檔案系統路徑名稱。請參閱整合檔案系統路徑名稱，取得詳細資訊。

記錄層次存取類別會使用下列項目：

- 用來說明資料庫檔案記錄的 RecordFormat 類別
- 提供資料庫檔案記錄的存取權的 Record 類別
- LineDataRecordWriter 類別以行式資料格式寫入記錄

這些類別均會在資料轉換一節中說明。

範例

- 循序存取範例說明如何依序存取 iSeries 檔。
- 讀取檔案範例說明如何使用記錄層次存取類別，讀取 iSeries 檔。
- 索引檔範例說明如何使用記錄層次存取類別，按照索引從 iSeries 檔中讀取記錄。

AS400File:

AS400File 類別會提供執行下列事項的方法：

- 建立與刪除伺服器實體檔案及成員
- 在伺服器檔案中讀取及寫入記錄
- 鎖定檔案進行不同類型的存取
- 使用記錄區塊改善效能
- 在已開啓的伺服器檔案內設定游標位置
- 管理確定控制活動

KeyedFile:

KeyedFile 類別給予 Java 程式在伺服器上索引存取檔案的權限。索引存取表示 Java 程式可藉由指定索引來存取檔案的記錄。按照索引來定位游標、讀取、更新以及刪除記錄的方法。

若要定位游標，請使用下列方法：

- positionCursor(Object[]) - 將游標設定到具有指定之索引的第一筆記錄。
- positionCursorAfter(Object[]) - 將游標設定到具有指定之索引的第一筆記錄之後。
- positionCursorBefore(Object[]) - 將游標設定到具有指定之索引的第一筆記錄之前。

若要刪除記錄，請使用下面方法：

- deleteRecord(Object[]) - 刪除具有指定之索引的的第一筆記錄。

讀取方法如下：

- read(Object[]) - 讀取具有指定索引的第一筆記錄
- readAfter(Object[]) - 讀取具有指定之索引的第一筆記錄之後的記錄。
- readBefore(Object[]) - 讀取具有指定之索引的第一筆記錄之前的記錄。
- readNextEqual() - 讀取索引符合指定之索引的下一筆記錄。將從現行游標位置後的記錄開始搜尋。
- readPreviousEqual() - 讀取索引符合指定之索引的前一筆記錄。將從現行游標位置前的記錄開始搜尋。

若要更新記錄，請使用下面方法：

- update(Object[]) - 更新具有指定之索引的記錄。

按照索引來定位、讀取以及更新時，也提供方法來指定搜尋準則。有效搜尋標準值如下：

- Equal - 尋找索引符合指定之索引的第一筆記錄。
- Less than - 尋找最後一筆記錄，其索引在檔案之索引次序的指定索引之前。
- Less than or equal - 尋找索引符合指定之索引的第一筆記錄。如果沒有記錄符合指定索引，請按照檔案的索引順序來尋找其索引位於指定索引前面的最後一筆記錄。
- Greater than - 尋找第一筆記錄，其索引在檔案之索引次序的指定索引之後。
- Greater than or equal - 尋找索引符合指定索引的第一筆記錄。如果沒有記錄符合指定索引，請按照檔案的索引順序來尋找其索引在指定索引後面的第一筆記錄。

KeyedFile 是 AS400File 的子類別；AS400File 中所有的方法皆可用於 KeyedFile。

指定索引

KeyedFile 物件的索引以「Java 物件」的陣列表示，其類型及次序對應到索引欄位的類型及次序 (由檔案的 RecordFormat 物件指定)。

下面範例說明如何指定 KeyedFile 物件的索引。

```
            // Specify the key for a file whose key fields, in order,
            // are:
            //    CUSTNAME    CHAR(10)
            //    CUSTNUM     BINARY(9)
            //    CUSTADDR    CHAR(100)VARLEN()
            // Note that the last field is a variable-length field.
    Object[] theKey = new Object[3];
    theKey[0] = "John Doe";
    theKey[1] = new Integer(445123);
    theKey[2] = "2227 John Doe Lane, ANYTOWN, NY 11199";
```

KeyedFile 物件接受部分索引以及完整索引。然而，必須依照順序指定索引欄位值。

例如：

```
            // Specify a partial key for a file whose key fields,
            // in order, are:
            //    CUSTNAME    CHAR(10)
            //    CUSTNUM     BINARY(9)
            //    CUSTADDR    CHAR(100)VARLEN()
    Object[] partialKey = new Object[2];
    partialKey[0] = "John Doe";
    partialKey[1] = new Integer(445123);

            // Example of an INVALID partial key
    Object[] INVALIDPartialKey = new Object[2];
    INVALIDPartialKey[0] = new Integer(445123);
    INVALIDPartialKey[1] = "2227 John Doe Lane, ANYTOWN, NY 11199";
```

不支援空字元索引和空字元索引欄位。

記錄的索引欄位值可透過 getKeyFields() 方法，從檔案的 Record 物件取得。

下面範例說明如何按照索引從檔案讀取。

```
                                // Create an AS400 object, the file exists on this
                                // server.
    AS400 sys = new AS400("mySystem.myCompany.com");

            // Create a file object that represents the file
    KeyedFile myFile = new KeyedFile(sys, "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/%FILE%.MBR");
            // Assume that the AS400FileRecordDescription class
            // was used to generate the code for a subclass of
            // RecordFormat that represents the record format
            // of file MYFILE in library MYLIB.  The code was
            // compiled and is available for use by the Java program.
    RecordFormat recordFormat = new MYKEYEDFILEFormat();

            // Set the record format for myFile.  This must
            // be done before invoking open()
    myFile.setRecordFormat(recordFormat);

            // Open the file.
    myFile.open(AS400File.READ_WRITE, 0, AS400File.COMMIT_LOCK_LEVEL_NONE);

            // The record format for the file contains
            // four key fields, CUSTNUM, CUSTNAME, PARTNUM
            // and ORDNUM in that order.
            // The partialKey will contain 2 key field
            // values.  Because the key field values must be
            // in order, the partialKey will consist of values for
            // CUSTNUM and CUSTNAME.
    Object[] partialKey = new Object[2];
    partialKey[0] = new Integer(1);
    partialKey[1] = "John Doe";

            // Read the first record matching partialKey
    Record keyedRecord = myFile.read(partialKey);

            // If the record was not found, null is returned.
    if (keyedRecord != null)
    { // Found the record for John Doe, print out the info.
      System.out.println("Information for customer " + (String)partialKey[1] + ":");
      System.out.println(keyedRecord);
    }

                    ....
```

```
        // Close the file since I am done using it
    myFile.close();

        // Disconnect since I am done using record-level access

sys.disconnectService(AS400.RECORDACCESS);
```

SequentialFile:

SequentialFile 類別給予 Java 程式按照記錄號碼存取伺服器上之檔案的權限。按照記錄號碼來定位游標、讀取、更新以及刪除記錄的方法。

若要定位游標，請使用下列方法：

- positionCursor(int) - 將游標設定到具有指定之記錄編號的記錄處。
- positionCursorAfter(int) - 將游標設定到指定之記錄編號的記錄之後。
- positionCursorBefore(int) - 將游標設定指定之記錄編號的記錄之前。

若要刪除記錄，請使用下面方法：

- deleteRecord(int) - 刪除具有指定之記錄編號的記錄。

若要讀取記錄，請使用下列方法：

- read(int) - 讀取具有指定之記錄編號的記錄。
- readAfter(int) - 讀取指定之記錄編號之後的記錄。
- readBefore(int) - 讀取指定之記錄編號之前的記錄。

若要更新記錄，請使用下面方法：

- update(int) - 更新具有指定之記錄編號的記錄。

SequentialFile 是 AS400File 的次類別；AS400File 中的全部方法皆可用於 SequentialFile。

下面範例說明如何使用 SequentialFile 類別：

```
                    // Create an AS400 object, the file exists on this
                    // server.
    AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a file object that represents the file
    SequentialFile myFile = new SequentialFile(sys, "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/%FILE%.MBR");

                    // Assume that the AS400FileRecordDescription class
                    // was used to generate the code for a subclass of
                    // RecordFormat that represents the record format
                    // of file MYFILE in library MYLIB.  The code was
                    // compiled and is available for use by the Java program.
    RecordFormat recordFormat = new MYFILEFormat();

                    // Set the record format for myFile.  This must
                    // be done before invoking open()
    myFile.setRecordFormat(recordFormat);

                    // Open the file.
    myFile.open(AS400File.READ_WRITE, 0, AS400File.COMMIT_LOCK_LEVEL_NONE);

                    // Delete record number 2.
    myFile.delete(2);

                    // Read record number 5 and update it
```

```
    Record updateRec = myFile.read(5);
    updateRec.setField("CUSTNAME", newName);

                      // Use the base class' update() method since I am
                      // already positioned on the record.
    myFile.update(updateRec);

                      // Update record number 7
    updateRec.setField("CUSTNAME", nextNewName);
    updateRec.setField("CUSTNUM", new Integer(7));
    myFile.update(7, updateRec);

                    ....

                      // Close the file since I am done using it
    myFile.close();

                      // Disconnect since I am done using record-level access

sys.disconnectService(AS400.RECORDACCESS);
```

AS400FileRecordDescription:

AS400FileRecordDescription 類別提供方法來擷取伺服器上某檔案的記錄格式。此類別所提供的方法，可用來爲 RecordFormat 的次類別建立 Java 原始程式碼以及傳回 RecordFormat 物件，以說明伺服器上使用者指定之實體或邏輯檔案的記錄格式。設定記錄格式時，這些方法的輸出可作爲 AS400File 物件的輸入。

當此檔案已存在於伺服器時，建議一律使用 AS400FileRecordDescription 類別來產生 RecordFormat 物件。

註: AS400FileRecordDescription 類別不會擷取檔案的整個記錄格式。僅擷取足夠資訊來描述構成檔案的記錄的內容。不會擷取如直欄標頭、別名及參照欄位等資訊。因此，所擷取的記錄格式不足以用來建立一個檔案，因爲其記錄格式同於從其中擷取格式的檔案。

為 RecordFormat 的次類別建立 Java 原始程式碼，以代表伺服器上的檔案記錄格式

createRecordFormatSource() 方法可爲 RecordFormat 類別的次類別建立 Java 來源檔。應用程式或 applet 可以編譯和使用這些檔案作爲 AS400File.setRecordFormat() 方法的輸入。

createRecordFormatSource() 方法應該作爲一種開發時間工具使用，來擷取伺服器上現有的檔案的記錄格式。此方法可讓 RecordFormat 類別之次類別的原始程式建立一次，必要時加以修改及編譯，然後供多個存取伺服器上相同檔案的 Java 程式使用。因爲此方法會在本端系統中建立檔案，所以只有 Java 應用程式能夠使用本方法。然而，您可編譯此輸出 (Java 原始程式碼)，然後同樣地由 Java 應用程式及 Applet 使用。

註: 此方法會覆寫與要建立之 Java 來源檔同名的檔案。

範例 1：下面範例說明如何使用 createRecordFormatSource() 方法：

```
                      // Create an AS400 object, the file exists on this
                      // server.
    AS400 sys = new AS400("mySystem.myCompany.com");

                      // Create an AS400FileRecordDescription object that represents the file
    AS400FileRecordDescription myFile = new AS400FileRecordDescription(sys,
                                        "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE");
                      // Create the Java source file in the current working directory.
                      // Specify "package com.myCompany.myProduct;" for the
                      // package statement in the source since I will ship the class
                      // as part of my product.
    myFile.createRecordFormatSource(null, "com.myCompany.myProduct");

                      // Assuming that the format name for file MYFILE is FILE1, the
```

```
                    // file FILE1Format.java will be created in the current working directory.
                    // It will overwrite any file by the same name.  The name of the class
                    // will be FILE1Format.  The class will extend from RecordFormat.
```

範例 2：編譯上述建立的檔案 FILE1Format.java，然後以下列方式使用此檔案：

```
                    // Create an AS400 object, the file exists on this
                    // server.
    AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create an AS400File object that represents the file
    SequentialFile myFile = new SequentialFile(sys, "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE");

                    // Set the record format
                    // This assumes that import.com.myCompany.myProduct.FILE1Format;
                    // has been done.

    myFile.setRecordFormat(new FILE1Format());

                    // Open the file and read from it
                    ....

                    // Close the file since I am done using it
    myFile.close();

                    // Disconnect since I am done using record-level access

sys.disconnectService(AS400.RECORDACCESS);
```

建立 RecordFormat 物件以代表伺服器上的檔案記錄格式

retrieveRecordFormat() 方法傳回 RecordFormat 物件的陣列，這些物件代表伺服器上現有的檔案的記錄格式。一般而言，在陣列中只傳回一個 RecordFormat 物件。當被擷取記錄格式的檔案是多重格式邏輯檔案時，會傳回一個以上的 RecordFormat 物件。使用此方法，在執行時間動態擷取伺服器上現有的檔案的記錄格式。然後可以使用 RecordFormat 物件作爲 AS400File.setRecordFormat() 方法的輸入。

下面範例說明如何使用 retrieveRecordFormat() 方法：

```
                    // Create an AS400 object, the file exists on this
                    // server.
    AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create an AS400FileRecordDescription object that represents the file
    AS400FileRecordDescription myFile = new AS400FileRecordDescription(sys,
                                    "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE");
                    // Retrieve the record format for the file
    RecordFormat[] format = myFile.retrieveRecordFormat();

                    // Create an AS400File object that represents the file
    SequentialFile myFile = new SequentialFile(sys, "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE");

                    // Set the record format
    myFile.setRecordFormat(format[0]);

                    // Open the file and read from it
                    ....

                    // Close the file since I am done using it
    myFile.close();

                    // Disconnect since I am done using record-level access
    sys.disconnectService(AS400.RECORDACCESS);
```

建立與刪除檔案及成員:

可藉由指定記錄長度、現有的伺服器資料說明規格 (DDS) 原始檔或 RecordFormat 物件，以建立伺服器上的實體檔案。

建立檔案並指定記錄長度之後，可建立資料檔或原始檔。此方法設定物件的記錄格式。 不要對此物件呼叫 setRecordFormat() 方法。

一個資料檔有一個欄位。欄位名稱是檔案的名稱，欄位類型是字元類型，欄位長度是在 create方法中指定的長度。

原始檔有三個欄位：

- 欄位 SRCSEQ 是 ZONED DECIMAL (6,2)
- 欄位 SRCDAT 是 ZONED DECIMAL (6,0)
- SRCDTA 是一個字元欄位，其長度是在 create 方法上指定的長度減去 12。

下列範例說明如何建立檔案與成員。

範例 1：建立具有 128 位元組記錄的資料檔：

```
                    // Create an AS400 object, the file
                    // will be created on this server.
     AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a file object that represents the file
    SequentialFile newFile = new SequentialFile(sys, "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/%FILE%.MBR");

                    // Create the file
    newFile.create(128, "*DATA", "Data file with a 128 byte record");

                    // Open the file for writing only.
                    // Note: The record format for the file
                    // has already been set by create()
    newFile.open(AS400File.WRITE_ONLY, 0, AS400File.COMMIT_LOCK_LEVEL_NONE);

                    // Write a record to the file.  Because the record
                    // format was set on the create(), getRecordFormat()
                    // can be called to get a record properly formatted
                    // for this file.
    Record writeRec = newFile.getRecordFormat().getNewRecord();
    writeRec.setField(0, "Record one");
    newFile.write(writeRec);

                  ....

                    // Close the file since I am done using it
    newFile.close();
                    // Disconnect since I am done using
                    // record-level access

sys.disconnectService(AS400.RECORDACCESS);
```

範例 2：建立一個指定現存 DDS 原始檔的檔案， 而且 DDS 原始檔是以 create() 方法來指定。開啓檔案之前， 必須使用 setRecordFormat() 方法設定檔案的記錄格式。例如：

```
                                        // Create an AS400 object, the
                    // file will be created on this server.
    AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create QSYSObjectPathName objects for
                    // both the new file and the DDS file.
    QSYSObjectPathName file    = new QSYSObjectPathName("MYLIB", "MYFILE", "FILE", "MBR");
    QSYSObjectPathName ddsFile = new QSYSObjectPathName("MYLIB", "DDSFILE", "FILE", "MBR");
```

```
                    // Create a file object that represents the file
     SequentialFile newFile = new SequentialFile(sys, file);

                    // Create the file
     newFile.create(ddsFile,  "File created using DDSFile description");

                    // Set the record format for the file
                    // by retrieving it from the server.
     newFile.setRecordFormat(new AS400FileRecordDescription(sys,
     newFile.getPath()).retrieveRecordFormat()[0]);

                    // Open the file for writing
     newFile.open(AS400File.WRITE_ONLY, 0, AS400File.COMMIT_LOCK_LEVEL_NONE);

                    // Write a record to the file.  The getRecordFormat()
                    // method followed by the getNewRecord() method is used to get
               // a default record for the file.
     Record writeRec = newFile.getRecordFormat().getNewRecord();
     newFile.write(writeRec);

                  ....

                    // Close the file since I am done using it
     newFile.close();
                    // Disconnect since I am done using
                    // record-level access

sys.disconnectService(AS400.RECORDACCESS);
```

範例 3：當建立一個指定 RecordFormat 物件的檔案時， 會在 create() 方法中指定 RecordFormat 物件。此方法設定物件的記錄格式。不可對此物件呼叫 setRecordFormat() 方法。

```
                                        // Create an AS400 object, the file will be created
                    // on this server.
     AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a file object that represents the file
     SequentialFile newFile = new SequentialFile(sys, "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/%FILE%.MBR");

                   // Retrieve the record format from an existing file
     RecordFormat recordFormat = new AS400FileRecordDescription(sys,
     "/QSYS.LIB/MYLIB.LIB/EXISTING.FILE/MBR1.MBR").retrieveRecordFormat()[0];

                    // Create the file
     newFile.create(recordFormat,  "File created using record format object");

                    // Open the file for writing only.
                    // Note: The record format for the file
                    // has already been set by create()
     newFile.open(AS400File.WRITE_ONLY, 0, AS400File.COMMIT_LOCK_LEVEL_NONE);

                    // Write a record to the file.  The recordFormat
                    // object is used to get a default record
                    // properly formatted for the file.
     Record writeRec = recordFormat.getNewRecord();
     newFile.write(writeRec);

                  ....

                    // Close the file since I am done using it
     newFile.close();
                    // Disconnect since I am done using
                    // record-level access

sys.disconnectService(AS400.RECORDACCESS);
```

刪除檔案與成員時，請使用這些方法：

- 使用 delete() 方法，以刪除伺服器檔案與其所有成員。
- 使用 deleteMember() 方法，僅刪除檔案的某一成員。

使用 addPhysicalFileMember() 方法，將成員新增到檔案。

讀取及寫入記錄:

您可使用 AS400File 類別，以讀取、寫入、更新及刪除伺服器上檔案中的記錄。可透過 Record 類別存取記錄，由 RecordFormat 類別所說明。開啓檔案之前， 必須透過 setRecordFormat() 方法設定記錄格式， 除非由其中一個 create() 方法 (設定物件的記錄格式) 建立檔案 (而中間沒有 close())。

使用 read() 方法來從檔案中讀取記錄。提供一些方法來執行下列動作：

- read() - 讀取位於現行游標位置的記錄
- readFirst() - 讀取檔案的第一筆記錄
- readLast() - 讀取檔案的最後一筆記錄
- readNext() - 讀取檔案的下一筆記錄
- readPrevious() - 讀取檔案中的前一筆記錄

下面範例說明如何使用 readNext() 方法：

```
                    // Create an AS400 object, the file exists on this
                    // server.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a file object that represents the file
SequentialFile myFile = new SequentialFile(sys, "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/%FILE%.MBR");

                    // Assume that the AS400FileRecordDescription class
                    // was used to generate the code for a subclass of
                    // RecordFormat that represents the record format
                    // of file MYFILE in library MYLIB.  The code was
                    // compiled and is available for use by the Java
                    // program.
RecordFormat recordFormat = new MYFILEFormat();

                    // Set the record format for myFile.  This must
                    // be done before invoking open()
myFile.setRecordFormat(recordFormat);

                    // Open the file.
myFile.open(AS400File.READ_ONLY, 0, AS400File.COMMIT_LOCK_LEVEL_NONE);

                    // Read each record in the file writing field
                    // CUSTNAME to System.out
System.out.println("             CUSTOMER LIST");
System.out.println("_____________________________________________");

Record record = myFile.readNext();
while(record != null)
{
  System.out.println(record.getField("CUSTNAME"));
  record = myFile.readNext();
}

                ....

                    // Close the file since I am done using it
myFile.close();
```

```
                        // Disconnect since I am done using
                        // record-level access.

sys.disconnectService(AS400.RECORDACCESS);
```

使用 update() 方法，以更新游標位置處的記錄。

例如：

```
                        // Create an AS400 object, the file exists on this
                        // server.
     AS400 sys = new AS400("mySystem.myCompany.com");

                        // Create a file object that represents the file
     SequentialFile myFile = new SequentialFile(sys, "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/%FILE%.MBR");

                        // Assume that the AS400FileRecordDescription class
                        // was used to generate the code for a subclass of
                        // RecordFormat that represents the record format
                        // of file MYFILE in library MYLIB.  The code was
                        // compiled and is available for use by the Java program.
     RecordFormat recordFormat = new MYFILEFormat();

                        // Set the record format for myFile.  This must
                        // be done prior to invoking open()
     myFile.setRecordFormat(recordFormat);

                        // Open the file for updating
     myFile.open(AS400File.READ_WRITE, 0, AS400File.COMMIT_LOCK_LEVEL_NONE);

                        // Update the first record in the file.  Assume
                        // that newName is a String with the new name for
                        // CUSTNAME
     Record updateRec = myFile.readFirst();
     updateRec.setField("CUSTNAME", newName);
     myFile.update(updateRec);

                    ....

                        // Close the file since I am done using it
     myFile.close();

                        // Disconnect since I am done using record-level access

sys.disconnectService(AS400.RECORDACCESS);
```

使用 write() 方法，將記錄附加到檔案尾端。單一記錄或記錄陣列可附加至檔案。

使用 deleteCurrentRecord() 方法，刪除游標位置處的記錄。

鎖定檔案:

Java 程式可以鎖定檔案，以防止當第一個 Java 程式正在使用檔案時，有其他的使用者存取該檔案。鎖定型類如下：

- 讀取/專用鎖定 - 現行 Java 程式讀取記錄時，其他程式無法存取此檔案。
- 讀取/容許共用讀取鎖定 - 現行 Java 程式讀取記錄時，其他程式可讀取檔案的記錄。
- 讀取/容許共用寫入鎖定 - 現行 Java 程式讀取記錄時，其他程式可變更檔案。
- 寫入/專用鎖定 - 現行 Java 程式變更檔案時，其他程式無法存取檔案。
- 寫入/容許共用讀取鎖定 - 現行 Java 程式變更檔案時，其他程式可讀取檔案的記錄。
- 寫入/容許共用寫入鎖定 - 現行 Java 程式變更檔案時，其他程式可變更檔案。

爲了放棄透過 lock() 方法所取得的鎖定，Java 程式會啓動 releaseExplicitLocks() 方法。

使用記錄區塊:

AS400File 類別會使用記錄區塊來改善效能：

- 如果此檔案開啓爲唯讀存取，則在 Java 程式讀取記錄時，會讀取記錄區塊。區塊可增進效能，因爲不必存取伺服器即可處理後續讀取要求。讀取單一記錄與讀取數筆記錄在效能上幾乎無差別。如果可從用戶端中置於快速記憶體中的記錄區塊伺服記錄，便能夠大幅增進效能。

  開啓檔案之後可設定在每一個區塊內要讀取的記錄數。例如：

```
                    // Create an AS400 object, the file exists on this
                    // server.
  AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a file object that represents the file
  SequentialFile myFile = new SequentialFile(sys, "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/%FILE%.MBR");

                    // Assume that the AS400FileRecordDescription class
                    // was used to generate the code for a subclass of
                    // RecordFormat that represents the record format
                    // of file MYFILE in library MYLIB.  The code was
                    // compiled and is available for use by the Java
                    // program.
  RecordFormat recordFormat = new MYFILEFormat();

                    // Set the record format for myFile.  This must
                    // be done before invoking open()
  myFile.setRecordFormat(recordFormat);

                    // Open the file.  Specify a blocking factor of 50.
  int blockingFactor = 50;
  myFile.open(AS400File.READ_ONLY, blockingFactor, AS400File.COMMIT_LOCK_LEVEL_NONE);

                    // Read the first record of the file.  Because
                    // a blocking factor was specified, 50 records
                    // are retrieved during this read() invocation.
  Record record = myFile.readFirst();
  for (int i = 1; i < 50 && record != null; i++)
  {

    // The records read in this loop will be served out of the block of
    // records cached on the client.
    record = myFile.readNext();
  }
                  ....

                    // Close the file since I am done using it
  myFile.close();

                    // Disconnect since I am done using
                    // record-level access

sys.disconnectService(AS400.RECORDACCESS);
```

- 如果此檔案開啓爲唯寫存取，則區塊因數指出，當呼叫 write(Record[]) 方法時一次有多少筆記錄寫入此檔案。

  例如：

```
                    // Create an AS400 object, the file exists on this
                    // server.
  AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a file object that represents the file
  SequentialFile myFile = new SequentialFile(sys, "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/%FILE%.MBR");
```

```
                    // Assume that the AS400FileRecordDescription class
                    // was used to generate the code for a subclass of
                    // RecordFormat that represents the record format
                    // of file MYFILE in library MYLIB.  The code was
                    // compiled and is available for use by the Java
                    // program.
RecordFormat recordFormat = new MYFILEFormat();

                    // Set the record format for myFile.  This must
                    // be done prior to invoking open()
myFile.setRecordFormat(recordFormat);

                    // Open the file.  Specify a blocking factor of 50.
int blockingFactor = 50;
myFile.open(AS400File.WRITE_ONLY, blockingFactor, AS400File.COMMIT_LOCK_LEVEL_NONE);

                    // Create an array of records to write to the file
Record[] records = new Record[100];
for (int i = 0; i < 100; i++)
{
                    // Assume the file has two fields,
                    // CUSTNAME and CUSTNUM
  records[i] = recordFormat.getNewRecord();
  records[i].setField("CUSTNAME", "Customer " + String.valueOf(i));
  records[i].setField("CUSTNUM", new Integer(i));
}

                    // Write the records to the file.  Because the
                    // blocking factor is 50, only two trips to the
                    // server are made with each trip writing 50 records
myFile.write(records);

                  ....

                    // Close the file since I am done using it
myFile.close();

                    // Disconnect since I am done using
                    // record-level access
sys.disconnectService(AS400.RECORDACCESS);
```

- 如果此檔案是開啓爲讀寫存取，則不會執行區塊化。系統不處理使用 open() 方法指定的任何區塊因數。

設定游標位置:

一個開啓檔案具有一個游標。游標指向要讀取、更新或刪除的記錄。最先開啓某檔案之後，游標會指向該檔案的開頭。檔案的開頭是在第一筆記錄前面。使用下列方法來設定游標位置：

- positionCursorAfterLast() - 設定游標在最後一個記錄之後。這項方法讓 Java 程式得以使用 readPrevious() 方法來存取檔案中的記錄。
- positionCursorBeforeFirst() - 設定游標在第一個記錄之前。這項方法讓 Java 程式得以使用 readNext() 方法來存取檔案中的記錄。
- positionCursorToFirst() - 設定游標在第一個記錄處。
- positionCursorToLast() - 設定游標在最後一個記錄處。
- positionCursorToNext() - 將游標移到下一個記錄。
- positionCursorToPrevious() - 將游標移到前一個記錄。

下面範例說明如何使用 positionCursorToFirst() 方法來定位游標。

```
                    // Create an AS400 object, the file exists on this
                    // server.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a file object that represents the file
SequentialFile myFile = new SequentialFile(sys, "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/%FILE%.MBR");

                    // Assume that the AS400FileRecordDescription class
                    // was used to generate the code for a subclass of
                    // RecordFormat that represents the record format
                    // of file MYFILE in library MYLIB.  The code was
                    // compiled and is available for use by the Java
                    // program.
RecordFormat recordFormat = new MYFILEFormat();

                    // Set the record format for myFile.  This must
                    // be done before invoking open()
myFile.setRecordFormat(recordFormat);

                    // Open the file.
myFile.open(AS400File.READ_WRITE, 1, AS400File.COMMIT_LOCK_LEVEL_NONE);

                    // I want to delete the first record of the file.
myFile.positionCursorToFirst();
myFile.deleteCurrentRecord();

                  ....

                    // Close the file since I am done using it
myFile.close();

                    // Disconnect since I am done using
                    // record-level access
sys.disconnectService(AS400.RECORDACCESS);
```

確定控制:

透過確定控制，Java 程式在變更檔案時將可取得進一步的控制。開啓確定控制之後，會擱置檔案的異動，直到確定或回轉檔案的異動爲止。一經確定之後，全部變更會存放到檔案。如果回轉，則會捨棄全部變更。根據使用 open() 方法所指定的確定控制鎖定層次而定，異動可能會變更現存記錄、新增記錄、刪除記錄，甚至讀取記錄。

確定控制的層次如下：

- 全部 - 確定或回轉異動之前，將鎖定在此檔案中存取的每筆記錄。
- 變更 - 確定或回轉異動之前，將鎖定在此檔案中更新、新增以及刪除的記錄。
- 游標穩定性 - 確定或回轉異動之前，將鎖定在此檔案中更新、新增以及刪除的記錄。直到存取另一筆記錄之前，會鎖定已存取但未變更的記錄。
- 無 - 在此檔案中沒有確定控制。變更會立即存放到檔案而且無法回轉。

您可使用 startCommitmentControl() 方法以啓動確定控制。確定控制應用於 AS400 **連接**。針對某連接啓動確定控制之後，從啓動確定控制開始，它便應用於在該連接之下開啓的全部檔案。啓動確定控制之前開啓的檔案不在確定控制之下。個別檔案的確定控制層次指定於 open() 方法上。指定 COMMIT_LOCK_LEVEL_DEFAULT，以使用在 startCommitmentControl() 方法上所指定的相同確定控制層次。

例如：

```
                    // Create an AS400 object, the files exist on this
                    // server.
AS400 sys = new AS400("mySystem.myCompany.com");
```

```
                    // Create three file objects
    SequentialFile myFile = new SequentialFile(sys, "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/%FILE%.MBR");
    SequentialFile yourFile = new SequentialFile(sys, "/QSYS.LIB/YOURLIB.LIB/YOURFILE.FILE/%FILE%.MBR");
    SequentialFile ourFile = new SequentialFile(sys, "/QSYS.LIB/OURLIB.LIB/OURFILE.FILE/%FILE%.MBR");

                    // Open yourFile before starting commitment control
                    // No commitment control applies to this file.  The
                    // commit lock level parameter is ignored because
                    // commitment control is not started for the connection.
    yourFile.setRecordFormat(new YOURFILEFormat());
    yourFile.open(AS400File.READ_WRITE, 0, AS400File.COMMIT_LOCK_LEVEL_DEFAULT);

                    // Start commitment control for the connection.
                    // Note: Any of the three files might be used for
                    // this call to startCommitmentControl().
    myFile.startCommitmentControl(AS400File.COMMIT_LOCK_LEVEL_CHANGE);

                    // Open myFile and ourFile
    myFile.setRecordFormat(new MYFILEFormat());

                    // Use the same commit lock level as specified
                    // when commitment control was started
    myFile.open(AS400File.WRITE_ONLY, 0, COMMIT_LOCK_LEVEL_DEFAULT);

    ourFile.setRecordFormat(new OURFILEFormat());
                    // Specify a different commit lock level than
                    // when commitment control was started
    ourFile.open(AS400File.READ_WRITE, 0, COMMIT_LOCK_LEVEL_CURSOR_STABILITY);

                    // write and update records in all three files
                   ....

                    // Commit the changes for files myFile and ourFile.
                    // Note that the commit commits all changes for the connection
              // even though it is invoked on only one AS400File object.
    myFile.commit();
                    // Close the files
    myFile.close();
    yourFile.close();
    ourFile.close();

                    // End commitment control
              // This ends commitment control for the connection.
    ourFile.endCommitmentControl();

                    // Disconnect since I am done using record-level access

sys.disconnectService(AS400.RECORDACCESS);
```

commit() 方法確認自上一次確認**連線**範圍後的所有異動。rollback() 方法捨棄自上一次確認**連線**範圍後的所有異動。可透過 endCommitmentControl() 方法結束連線的確定控制。如果在呼叫 commit() 或 rollback() 方法之前結束檔案，則會回轉全部未確定的異動。在呼叫 endCommitmentControl() 方法之前，必須結束在確定控制下開啓的全部檔案。

下面範例說明如何啓動確定控制、確定或回轉功能，然後結束確定控制：

```
                    // ... assume the AS400 object and file have been
                    // instantiated.

                    // Start commitment control for *CHANGE
    aFile.startCommitmentControl(AS400File.COMMIT_LOCK_LEVEL_CHANGE);

                    // ... open the file and do several changes.  For
                    // example, update, add or delete records.

                    // Based on a flag either save or discard the
```

```
                    // transactions.
if (saveChanges)
   aFile.commit();
else
   aFile.rollback();

                    // Close the file
aFile.close();

                    // End commitment control for the connection.
aFile.endCommitmentControl();
```

ServiceProgramCall 類別

ServiceProgramCall 類別可讓您呼叫 iSeries 服務程式。ServiceProgramCall 是 ProgramCall 類別的次類別，該類別可呼叫 iSeries 程式。如果您要呼叫 iSeries 程式，請使用 ProgramCall 類別。

ServiceProgramCall 類別可讓您呼叫 iSeries 服務程式、經由輸入參數傳送資料到 iSeries 服務程式，並經由輸出參數存取 iSeries 服務程式傳回的資料。使用 ServiceProgramCall 會使 AS400 物件連接到 iSeries。有關管理連線的資訊，請參閱管理連線。

預設行爲是，即使在 Java 程式及服務程式位於同一伺服器上時，也可以讓服務程式在不同的伺服器工作中執行。您可以置換預設行爲，並使用繼承的 (來自 ProgramCall) setThreadSafe() 方法，讓服務程式在 Java 工作中執行。

使用 ServiceProgramCall 類別

要使用 ServiceProgramCall 類別，您必須確定遵循下列基本要求：

- 服務程式必須位於 iSeries 上
- 您最多只能傳送七個參數到服務程式
- 服務程式的回傳值爲取消或是數字。

使用 ProgramParameter 物件

ProgramParameter 類別使用 ServiceProgramCall 類別，以傳遞參數資料至 iSeries 服務程式，或自其中取得參數資料。輸入資料可以使用 setInputData() 傳送到 iSeries 服務程式。

您可以使用 setOutputDataLength() 要求要傳回的輸出資料數量。使用 getOutputData() 可以在服務程式結束執行之後，擷取輸出資料。除了資料本身以外，ServiceProgramCall 也需要知道如何傳送參數資料到服務程式。ProgramParameter 的 setParameterType() 方法可用來提供這個資訊。該類型可以指出以值傳送參數或以參照傳送參數。在任何一種情形下，資料都是自用戶端傳送到伺服器。一旦資料到達 iSeries，伺服器便會使用參數類型來正確地呼叫服務程式。

所有參數都是在位元組陣列的套表中。因此，若要在 iSeries 與 Java 格式之間轉換，您可以使用資料轉換及說明類別。

SystemStatus 類別

SystemStatus 類別可讓您擷取系統狀態資訊，以及擷取及變更系統儲存區資訊。SystemStatus 物件可讓您擷取包含於下列的系統狀態資訊：

- getUsersCurrentSignedOn()：傳回目前登入系統的使用者數
- getUsersTemporarilySignedOff()：傳回斷線的互動式作業數
- getDateAndTimeStatusGathered()：傳回收集系統狀態資訊時的日期與時間

- getJobsInSystem()：傳回目前正在執行的使用者與系統工作總數
- getBatchJobsRunning()：傳回目前在系統上執行的批次作業數
- getBatchJobsEnding()：傳回正在結束程序的批次作業數
- getSystemPools()：爲每個系統儲存區傳回包含 SystemPool 物件的細目

除了 SystemStatus 類別內的方法外，您也可以透過 SystemStatus 存取 SystemPool。SystemPool 可讓您取得系統儲存區的相關資訊及變更系統儲存區資訊。

範例

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

此範例顯示您如何以 SystemStatus 類別來使用快取：

```
AS400 system = new AS400("MyAS400");
SystemStatus status = new SystemStatus(system);

// Turn on caching. It is off by default.
status.setCaching(true);

// This will retrieve the value from the system.
// Every subsequent call will use the cached value
// instead of retrieving it from the system.
int jobs = status.getJobsInSystem();


// ... Perform other operations here ...


// This determines if caching is still enabled.
if (status.isCaching())
{
  // This will retrieve the value from the cache.
  jobs = status.getJobsInSystem();
}

// Go to the system next time, regardless if caching is enabled.
status.refreshCache();

// This will retrieve the value from the system.
jobs = status.getJobsInSystem();

// Turn off caching. Every subsequent call will go to the system.
status.setCaching(false);

// This will retrieve the value from the system.
jobs = status.getJobsInSystem();
```

SystemPool 類別:

SystemPool 類別可讓您擷取並變更系統儲存區資訊，包括如下：

- getPoolSize() 方法可以傳回儲存區大小，而 setPoolSize() 方法則是設定儲存區大小。
- getPoolName() 方法可以擷取儲存區名稱，而 setPoolName() 方法則是設定儲存區的名稱。
- getReservedSize() 方法可以傳回儲存體的數量，而這些儲存體位於保留給系統使用的儲存區中。
- getDescription() 方法可以傳回系統儲存區的說明。
- getMaximumActiveThreads() 方法可以傳回儲存區中，隨時可以啓動的緒數目上限。
- setMaximumFaults() 方法可以設定每秒錯誤 (faults-per-second) 最大值引導線，以用於此系統儲存區。
- setPriority() 方法可以設定此系統儲存區相對於其它系統儲存區的優先順序。

範例

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
//Create AS400 object.
AS400 as400 = new AS400("system name");

//Construct a system pool object.
SystemPool systemPool = new SystemPool(as400,"*SPOOL");

//Get system pool paging option
System.out.println("Paging option : "+systemPool.getPagingOption());
```

系統值

系統值類別可讓 Java 程式擷取並變更系統值及網路屬性。您也可以定義您自己的群組以包含您要的系統值。

SystemValue 物件主要包含下列資訊：

- 名稱
- 說明
- 版次
- 值

利用 SystemValue 類別，使用 getValue() 方法擷取單一系統值，並使用 setValue() 方式變更系統值。

您也可以擷取關於特定系統值的群組資訊：

- 若要擷取系統值所屬的系統定義之群組，請使用 getGroup() 方法。
- 若要擷取 SystemValue 物件所屬的使用者定義之群組 (如果有的話)，請使用 getGroupName() 以及 getGroupDescription() 方法。

每當第一次擷取系統值時，都會從 iSeries 擷取該值並進行快取。在後續的擷取中，將傳回快取的值。如果您要的是現行 iSeries 值而非快取值，您必須先執行 clear()，以清除目前的快取。

系統值清單

SystemValueList 代表指定 iSeries 伺服器上的系統值清單。這個清單將分成數個系統定義的群組，可讓 Java 程式一次存取某一部分的系統值。

系統值群組

SystemValueGroup 代表使用者定義的系統值與網路屬性集合。與其說是儲存區，不如說它是產生及維護唯一系統值集合的 Factory。

若要建立 SystemValueGroup，可以指定其中一個系統定義的群組 (SystemValueList 類別中的其中一個常數) 或指定一個系統值名稱陣列。

您可以使用 add() 方法，單獨新增系統值名稱於群組中。也可以使用 remove() 方法來移除系統值名稱。

在以必要的系統值名稱移入 SystemValueGroup 後，可以呼叫 getSystemValues() 方法，從群組取得實際的 SystemValue 物件。這樣的話，SystemValueGroup 物件會取得一組的系統值名稱並且產生一個 SystemValue 物件的「向量」，而它們都具有 SystemValueGroup 的系統、群組名稱以及群組說明。

若要同時重新整理所有 SystemValue 物件的「向量」，請使用 refresh() 方法。

使用 SystemValue 及 SystemValueList 類別的範例

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

下列範例將告訴您如何建立及擷取系統值：

```
//Create an AS400 object
 AS400 sys = new AS400("mySystem.myCompany.com");

//Create a system value representing the current second on the system.
SystemValue sysval = new SystemValue(sys, "QSECOND");


//Retrieve the value.
String second = (String)sysval.getValue();


//At this point QSECOND is cached. Clear the cache to retrieve the most
//up-to-date value from the system.
sysval.clear();
second = (String)sysval.getValue();

//Create a system value list.
SystemValueList list = new SystemValueList(sys);


//Retrieve all the of the date/time system values.
Vector vec = list.getGroup(SystemValueList.GROUP_DATTIM);


//Disconnect from the system.
sys.disconnectAllServices();
```

使用 SystemValueGroup 類別的範例

下列範例顯示如何建置一組系統值名稱以及維護它們的方法：

```
//Create an AS400 object
 AS400 sys = new AS400("mySystem.myCompany.com");

//Create a system value group initially representing all of the network attributes on the system.
String name = "My Group";
String description = "This is one of my system values.";
SystemValueGroup svGroup = new SystemValueGroup(sys, name, description, SystemValueList.GROUP_NET);

//Add some more system value names to the group and remove some we do not want.
svGroup.add("QDATE");
svGroup.add("QTIME");
svGroup.remove("NETSERVER");
svGroup.remove("SYSNAME");

//Obtain the actual SystemValue objects. They are returned inside a Vector.
Vector sysvals = svGroup.getSystemValues();

//You will notice that this is one of my system values.
SystemValue mySystemValue = (SystemValue)sysvals.elementAt(0);
System.out.println(mySystemValue.getName()+" - "+mySystemValue.getGroupDescription());

//We can add another SystemValue object from another system into the group.
AS400 sys2 = new AS400("otherSystem.myCompany.com");
SystemValue sv = new SystemValue(sys2, "QDATE");
sysvals.addElement(sv);

//Now refresh the entire group of system values all at once.
//It does not matter if some system values are from different iSeries servers.
//It does not matter if some system values were generated using SystemValueGroup and some were not.
```

```
SystemValueGroup.refresh(sysvals);

//Disconnect from the systems.
sys.disconnectAllServices();
sys2.disconnectAllServices();
```

Trace 類別

Trace 類別可讓 Java 程式記載追蹤點及診斷訊息。這個資訊可協助重新產生與診斷問題。

Trace 類別

註: 您也可以利用追蹤系統內容來設定追蹤。

Trace 類別會記載下列種類的資訊：

資訊種類	說明
轉換	記載 Unicode 與字碼頁之間的字集轉換。此種類僅適用於 IBM Toolbox for Java 類別。
資料串流	記載 iSeries 與 Java 程式之間流動的資料。此種類僅適用於 IBM Toolbox for Java 類別。
診斷	記載狀態資訊。
錯誤	記載其它引起異常情況的錯誤。
參考	追蹤程式的流程。
PCML	此種類是用來決定 PCML 如何解譯在伺服器之間往返傳送的資料。
Proxy	IBM Toolbox for Java 類別用此種類來記載用戶端與 proxy 伺服器之間流動的資料。
警告	記載程式可從其中恢復的錯誤的資訊。
全部	此種類可用來同時啓動或關閉以上所有種類的追蹤。追蹤資訊無法直接記錄到這個種類中。

IBM Toolbox for Java 類別也會使用追蹤種類。當 Java 程式啓用記載功能時，IBM Toolbox for Java 資訊會包含在應用程式所記載的資訊中。

您可以對單一種類或一組種類啓用追蹤功能。一旦選取了種類，請使用 setTraceOn 方法，來開啓及關閉追蹤。並使用 Log 方法將資料寫入日誌中。

您可以將不同元件的追蹤資料傳送到個別的日誌。在預設的情況下，追蹤資料會寫入預設日誌中。請使用元件追蹤，將特定應用程式追蹤資料寫入個別的日誌或標準輸出。藉由使用元件追蹤，您可以輕鬆地將特定應用程式的追蹤資料與其它資料分開。

過多的記載可能會影響效能。使用 isTraceOn 方法，來查詢追蹤的現行狀態。您的 Java 程式可以使用此方法，來決定在呼叫 log 方法之前，是否先建置追蹤記錄。在關閉記載功能時呼叫 log 方法不是一種錯誤，但會花費較長的時間。

預設值為將日誌資訊寫入至標準輸出中。若要將日誌重新導向至檔案，請從 Java 應用程式呼叫 setFileName() 方法。一般說來，這僅對 Java 應用程式有作用，因爲大多數瀏覽器並不會授予 Applet 本端檔案系統的寫入權。

就預設值而言，記載功能是關閉的。Java 程式爲使用者提供了一個開啓記載功能的方法，以便輕鬆地啓用記載功能。例如，應用程式可以剖析指令行參數，來指出要記載的資料種類。當需要日誌資訊時，使用者可以設定這個參數。

範例

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

下列範例說明如何使用 Trace 類別。

使用 setTraceOn() 並使用 log 方法將資料寫入日誌的範例

```
      // Enable diagnostic, information, and warning logging.
      Trace.setTraceDiagnosticOn(true);
      Trace.setTraceInformationOn(true);
      Trace.setTraceWarningOn(true);

      // Turn tracing on.
      Trace.setTraceOn(true);

      // ... At this point in the Java program,
      Trace.log(Trace.INFORMATION, "Just entered class xxx, method xxx");

      // Turning tracing off.
      Trace.setTraceOn(false);
```

範例：使用追蹤

在下列程式碼中，「方法 2 是使用追蹤的偏好方式。

```
      // Method 1 - build a trace record
      // then call the log method and let the trace class determine if the
      // data should be logged. This will work but will be slower than the
 // following code.
      String traceData = new String("Just entered class xxx, data = ");
     traceData = traceData + data + "state = " + state;
     Trace.log(Trace.INFORMATION, traceData);

      // Method 2 - check the log status before building the information to
      // log. This is faster when tracing
 if (Trace.isTraceOn() && Trace.isTraceInformationOn())
     {
        String traceData = new String("just entered class xxx, data = ");
        traceData = traceData + data + "state = " + state;
        Trace.log(Trace.INFORMATION, traceData);
     }
```

範例：使用元件追蹤

```
      // Create a component string. It is more efficient to create an
 // object than many String literals.
      String myComponent1 = "com.myCompany.xyzComponent";
      String myComponent2 = "com.myCompany.abcComponent";

     // Send IBM Toolbox for Java and the component trace data each to separate files.
     // The trace will contain all trace information, while each
     // component log file will only contain trace information specific to
     // that component.  If a Trace file is not specified, all trace data
     // will go to standard out with the component specified in front of
     // each trace message.

     // Trace.setFileName("c:\\bit.bucket");
     // Trace.setFileName(myComponent1, "c:\\Component1.log");
     // Trace.setFileName(myComponent2, "c:\\Component2.log");
```

```
Trace.setTraceOn(true);             // Turn trace on.
Trace.setTraceInformationOn(true);  // Enable information messages.

// Log component specific trace data or general IBM Toolbox for Java
// trace data.

Trace.setFileName("c:\\bit.bucket");
Trace.setFileName(myComponent1, "c:\\Component1.log");
```

使用者與群組

使用者與群組類別可讓您透過 Java 程式，取得 iSeries 伺服器上的使用者與使用者群組清單，以及每一位使用者的相關資訊。

註: IBM Toolbox for Java 同時具有資源類別，可提供同屬組織架構及一致的程式設計介面，用以處理不同的 iSeries 物件及清單。在您閱讀過存取套件和 resource 套件中各種類別的相關資訊後，即可針對您的應用程式選擇最適合的物件。用來處理與使用者相關的資源類別包括 RUser 和 RUserList。

您可以擷取某些使用者資訊，包括前次登入日期、狀態、上次變更密碼的日期、密碼到期日及使用者類別。當您存取 User 物件時，請使用 setSystem() 方法來設定系統名稱，並使用 setName() 方法來設定使用者名稱。執行這些步驟之後，您可以使用 loadUserInformation() 方法，從 iSeries 取得資訊。

UserGroup 物件代表一個特殊使用者，其使用者設定檔是群組設定檔。藉由使用 getMembers() 方法，即可傳回該群組的成員使用者清單。

Java 程式可以使用列舉方式重複傳回清單。所有列舉的元素皆爲 User 物件；例如：

```
 // Create an AS400 object.
AS400 system = new AS400 ("mySystem.myCompany.com");

// Create the UserList object.
UserList userList = new UserList (system);

// Get the list of all users and groups.
Enumeration enum = userList.getUsers ();

// Iterate through the list.
 while
(enum.hasMoreElements ())
{
    User u = (User) enum.nextElement ();
    System.out.println(u);
}
```

擷取使用者和群組的相關資訊

您可以使用 UserList 來取得下列各項的清單：

- 全部使用者和群組
- 僅群組
- 爲群組成員的所有使用者
- 非群組成員的所有使用者

UserList 物件中必須設定的唯一內容即是 AS400 物件，它代表將從其中擷取使用者清單的系統。

依據預設值，將傳回所有使用者。合併使用 setUserInfo() 和 setGroupInfo()，可以明確指定傳回的使用者。

範例：使用 UserList 列出給定群組中的所有使用者。

UserSpace 類別

UserSpace 類別代表伺服器上的使用者空間。必要的參數爲使用者空間的名稱，和代表擁有使用者空間的伺服器之 AS400 物件。存在於使用者空間中的方法將執行下列：

- 建立使用者空間
- 刪除使用者空間
- 從使用者空間讀取資料
- 寫入使用者空間。
- 取得使用者空間的屬性。Java 程式可以取得使用者空間的起始值、長度值及自動延伸屬性。
- 設定使用者空間的屬性。Java 程式可以設定使用者空間的起始值、長度值及自動延伸屬性。

UserSpace 物件需要程式的整合檔案系統路徑名稱。請參閱整合檔案系統路徑名稱，取得詳細資訊。

使用 UserSpace 類別會使得 AS400 物件連接到伺服器。有關管理連線的資訊，請參閱管理連線。

下面範例會建立一個使用者空間，然後將資料寫入其中。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a user space object.
UserSpace US = new UserSpace(sys,
          "/QSYS.LIB/MYLIB.LIB/MYSPACE.USRSPC");

                    // Use the create method to create the user space on
                    // the server.
US.create(10240,                          // The initial size is 10KB
          true,                           // Replace if the user space already exists
          " ",                            // No extended attribute
          (byte) 0x00,                    // The initial value is a null
          "Created by a Java program",    // The description of the user space
          "*USE");                        // Public has use authority to the user space


                    // Use the write method to write bytes to the user space.
US.write("Write this string to the user space.", 0);
```

Commtrace 類別

IBM Toolbox for Java Commtrace 類別可讓您的 Java 程式使用指定 LAN (乙太網路或記號環) 線路說明的通訊追蹤資料。commtrace 套裝軟體包含一個可用來當作獨立式公用程式執行的類別，以格式化通訊追蹤資料。

當您將 iSeries 伺服器的通訊追蹤傾出至串流檔時，該資訊就會以二進位格式儲存。commtrace 類別可讓您使用串流檔的各種元件。

註: 通訊追蹤檔可能含有機密的資訊，例如未解密的密碼。當通訊追蹤檔位於 iSeries 伺服器上時，只有具備 *SERVICE 特殊權限的使用者才能存取該項追蹤資料。如果您將該檔案移至用戶端，請確定您已經爲該檔案做好適當的防範。如需通訊追蹤的相關資訊，請參閱本頁底端的鏈結。

使用 commtrace 類別來執行下列工作：

- 格式化原始追蹤資料。
- 剖析任何資料以擷取您想要的資料。假設您是使用 commtrace 類別來格式化資料，您就可以對原始資料以及格式化的資料兩者進行剖析。

如需以視覺化呈現方式來瞭解 commtrace 類別如何代表通訊追蹤檔中的結構，請參閱下列頁面：

『Commtrace 模型』

commtrace 套裝軟體包含下列類別：

第 167 頁的『Format 與 FormatProperties 類別』： Format 類別能讀取通訊追蹤的原始資料以及格式化資料。FormatProperties 會設定 Format 物件的內容，例如，開始及結束時間、 IP 位址以及埠等等。

註: 您也可以將 Format 類別作爲獨立式程式來執行。

第 170 頁的『Prolog 類別』：從 iSeries 伺服器通訊追蹤的起始 256 位元組區段擷取資訊。

第 170 頁的『Frame 類別』：擷取通訊追蹤訊框的相關資訊。

第 171 頁的『LanHeader 類別』：從訊框開始處，曾經出現過的資料區段中擷取資訊。這個區段通常含有硬體特定的資訊，包括訊框的一般資訊，例如，訊框數、資料長度等等。

第 171 頁的『IPPacket 類別』：從封包中的資料擷取資訊。對於 commtrace 套裝軟體所支援的不同類型的資料包而言，IPPacket 是抽象的上層類別。

第 172 頁的『Header 類別』：從封包標頭及其相關資料擷取資訊。對於 commtrace 套裝軟體所支援的不同類型的封包標頭而言，Header 是抽象的上層類別。

com.ibm.as400.commtrace 套裝軟體中其餘的大部分類別，對您要使用的追蹤資料類型來說都是特定的。如需通訊追蹤以及所有 commtrace 類別的詳細資訊，請參閱下列頁面：

通訊追蹤

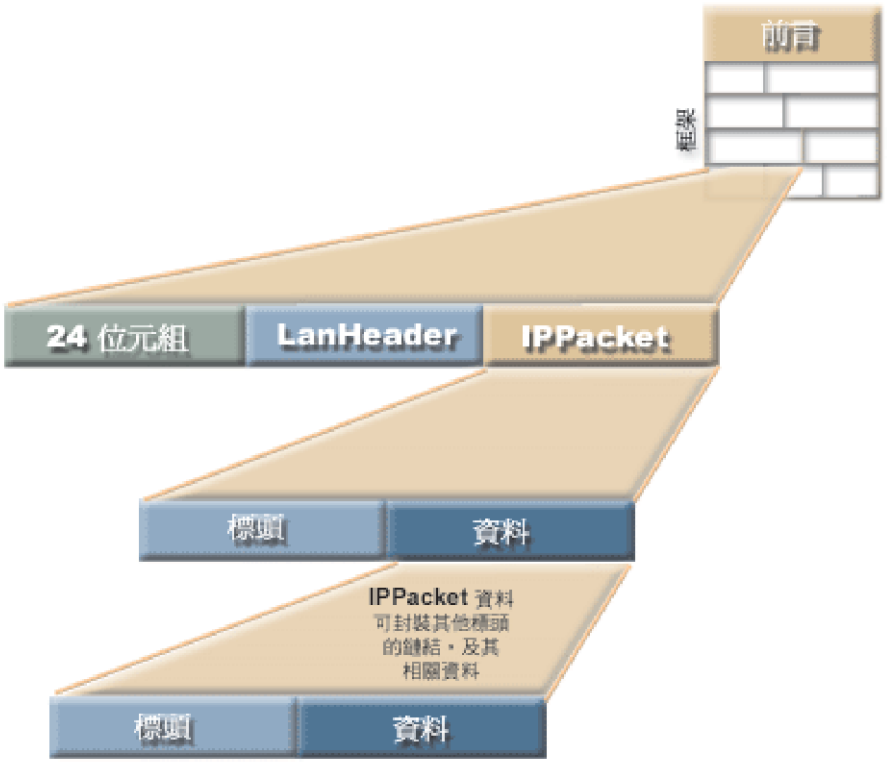
Commtrace 模型

本圖說明 commtrace 類別對應到通訊追蹤檔的方式。本圖也顯示 commtrace 類別針對通訊追蹤中的元件所使用之命名慣例。

前言：LAN 線路說明中通訊追蹤的起始 256 位元組區段。「前言」包含追蹤的一般資訊，例如開始與結束時間、收集的位元組數等等。

訊框：可變長度的記錄，其中包含通訊追蹤期間 iSeries 伺服器所傳輸的資料。

圖 1： Commtrace 模型



追蹤檔中的每一個訊框都含有兩個起始區段 (用來提供有關訊框內容的一般資訊) 以及 iSeries 伺服器在其本身與網路上不同的點之間所傳輸的封包。

資料最前面的 24 位元組區段包含有關訊框內容的一般資訊，例如訊框數以及資料長度。請使用 Frame 類別來處理這一資訊。

LanHeader：一個資料區段，它在每一個訊框中出現一次 (在資料的起始 24 位元組)，並含有隨後 IPacket 的一般資訊。

IPPacket：一個資料區段，它由一或多個標頭及其相關資料所組成。 IPPacket 代表於通訊追蹤期間，針對這個訊框，在網路上傳輸的所有資料封包。IPPacket 是一個抽象類別，因此您將會使用各種具體的子類別來處理封包中的標頭與資料。

標頭：IPPacket 開頭的資料區段， 該 IPPacket 說明後續的封包資料，並適時指向下一個標頭。 在 commtrace 套裝軟體中，Header 是一個抽象類別，因此您將會使用各種具體的子類別來處理資料。

圖 1 的長說明：Commtrace 模型 (rzahh587.gif):

位於 IBM Toolbox for Java：Commtrace 類別

本圖概略地說明 commtrace 類別對應到通訊追蹤檔的方式。

說明

此圖由下列項目所構成：

- 背景右側標示爲 'Prolog,' 的正方形影像。這個正方形分爲不同的列，代表用來作 LAN 線路說明之通訊追蹤的區段。影像左側垂直寫下來的字是 'Frames'

- 水平矩形前景左側的影像。這個矩形的各邊連接到背景中的主要正方形。該矩形代表了追蹤檔中的框架，分爲 3 個區段：
  - 綠色部分標示爲 24 位元組，代表包含有關框架內容一般資訊的資料區段。
  - 淺藍色部分標示爲 LanHeader，代表包含有關後接的 IPacket 一般資訊的資料區段。
  - 褐色部分標示爲 IPPacket，其代表由一或多個標頭及其相關資料所組成的資料區段。
- 第一個下面還有另一個水平矩形。該矩形的各邊都有一條線將它連接至 IPPacket。該矩形代表了 IPPacket 區段的明細版本，分成 2 個區段：
  - 淺藍色部分標示爲 Header，代表 IPPacket 開頭的資料區段
  - 深藍色部分標示爲 Data，代表封包資料並指向下一個標頭。
- 第三個水平矩形出現在深藍色 Data 區段的下方。 該矩形的各邊都有一條線將它連接至 Data 區段。該矩形代表了 IPPacket 的 2 個區段，並且註明「IPPacket 資訊會壓縮一系列其他標頭及其相關資料」。
-
  - 淺藍色部分標示爲 Header，代表 IPPacket 開頭的資料區段
  - 深藍色部分標示爲 Data，代表封包資料並指向下一個標頭。

追蹤檔 (背景影像) 中的框架 (第一個水平矩形) 包含了 2 個資料區段，24 位元組資料區段 (綠色) 以及靠近褐色封包的 (淺藍色) LanHeader。

IPPacket (第二個水平矩形) 會在一個框架中加以處理， 該框架由 (淺藍色) Header 以及依次指向另一個 Header 與 Data (第三個水平矩形) 的 (深藍色) Data 所組成。第二和第三個矩形透過其各邊的線條，從 Data 區段連接至 IPPacket 的背面，在通訊追蹤檔中，顯示爲標頭及其相關資料連續性的鏈結。

Format 與 FormatProperties 類別

Format 類別可用來作爲呼叫程式與追蹤訊框之間的介面。FormatProperties 類別可讓您設定與擷取某些內容，以決定 Format 物件在遇到追蹤「訊框」時的行爲。

Format 類別

使用 format 類別，讀取原始追蹤資料以及您已經用 commtrace 類別來格式化的資料。

註: 您無法用 commtrace 類別來讀取您使用「列印通訊追蹤 (PRTCMNTRC)」控制語言指令所格式化的通訊追蹤。

請使用 Format 類別來剖析並格式化追蹤中的資訊，然後將該項格式化的資訊傳送到檔案或列印裝置。此外，您可以建立能在獨立應用程式或瀏覽器中顯示資訊的圖形式前端。如果您只要選取特定資料，請使用 Format 類別，將該資訊提供給 Java 程式。例如，您可以使用 Format 類別，從追蹤當中讀取 IP 位址， 然後將這一資料運用到您的程式中。

Format 建構子能接受代表未格式化資料的引數，例如 IFSFileInputStream 物件、本機檔案或二進位追蹤檔。若要顯示您已經格式化的追蹤，請使用預設的 Format 建構子，然後再用 Format.openIFSFile() 或 Format.openLclFile() 來指定您要顯示的格式化檔案。

範例

下列範例將說明，如何顯示已儲存的追蹤或格式化二進位追蹤。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

範例：顯示已儲存的追蹤

```
Format fmt = new Format();
fmt.openLclFile("/path/to/file");

// Read the Prolog
System.out.println(fmt.getRecFromFile());
// The total number of records in the trace TCP and non-TCP
System.out.println("Total Records:" + fmt.getIntFromFile());
String rec;
// Read in records until we reach the end.
while((rec = fmt.getRecFromFile())!=null) {
System.out.println(rec);
```

Example: Formatting a binary trace

```
// Create a FormatProperties. By default display everything.
FormatProperties fmtprop = new FormatProperties();


Format fmt = new Format("/path/to/file");
// Sets the filtering properties for this format
fmt.setFilterProperties(fmtprop);
fmt.setOutFile("/path/to/output/file");
// Format the prolog
fmt.formatProlog();
// Format the trace and send data to the specified file
fmt.toLclBinFile();
```

Format 作為獨立式公用程式執行

您也可以將 Format 類別作爲獨立式公用程式來執行。如需詳細資訊，請參閱下列主題資訊：

Format 作爲獨立式程式執行

FormatProperties 類別

使用 FormatProperties 類別，指定並擷取 Format 物件的內容。換言之，當您使用 Format 類別來傳送資訊至檔案時，請使用 FormatProperties 類別來過濾您要傳送的資訊。

您可以透過這些內容來指定， Format 物件要如何處理它在通訊追蹤「訊框」中遇到的資訊。根據預設行爲，Format 物件會忽略那些您並未特別指定值的內容。

FormatProperties 類別提供您可以用來設定內容的常數。設定內容能讓 Format 物件確認您要使用的過濾器。例如，下列程式碼會設定 Format 物件來顯示進度對話框，而非廣播訊框：

```
FormatProperties prop = new FormatProperties();
prop.setProgress(FormatProperties.TRUE);
prop.setBroadcast(FormatProperties.NO);
```

大部分的內容都可讓 Format 物件作爲過濾器，供您設定以明確納入特定資料。當您設定了過濾器之後，Format 物件就只顯示那些符合過濾條件的資料。例如，下列程式碼會設定過濾器來顯示於特定的開始與結束時間之間所發生的訊框：

```
FormatProperties prop = new FormatProperties();
  // Set the filter to start and end times of 22 July, 2002,
  // 2:30 p.m. and 2:45 p.m. GMT.
  // The time is expressed as a Unix(TM) timestamp, which is
  // based on the standard epoch of 01/01/1970 at 00:00:00 GMT.
  prop.setStartTime("1027348200");
  prop.setEndTime("1027349100");
```

範例

下列範例將說明，如何利用眾多的 commtrace 類別，包括 Format 與 FormatProperties 類別，在您的監視器上顯示追蹤資訊：

第 173 頁的『範例：使用 commtrace 類別』

Javadoc 參考文件

有關 Format 與 FormatProperties 類別的詳細資訊，請參閱下列 Javadoc 參考文件：

Format

FormatProperties

Format 作為獨立式程式執行:

除了在您的 Java 程式中使用 Format 類別之外，您還可以將它作爲一個獨立式指令行公用程式來執行，以格式化通訊追蹤。該程式會將 IFSFileOutputStream 連接到特定的輸出檔，並將資料寫入該檔案。

將 Format 作爲獨立式公用程式執行，能夠讓您利用 iSeries 伺服器上的處理功能以及儲存空間來格式化檔案。

從指令行執行 Format

若要從指令行提示中執行 Format 公用程式，請使用下列指令：

```
java com.ibm.as400.commtrace.Format [options]
```

其中的 [options] 等於一或多個可用選項。選項包括：

- 要連接的系統
- 用於該系統的使用者 ID 及密碼
- 要剖析的通訊追蹤
- 用來儲存結果的檔案

如需可用選項的完整清單，請參閱下列資訊：

Format 類別的 Javadoc 參考文件

遠端執行 Format

若要從遠端來執行這個類別，請使用 JavaApplicationCall 類別：

```
// Construct a JavaApplicationCall object.
jaCall = new JavaApplicationCall(sys);
// Set the Java application you want to run.
jaCall.setJavaApplication("com.ibm.as400.util.commtrace.Format");
// Set the classpath environment variable used by the JVM on
// the server, so it can find the class to run.
jaCall.setClassPath("/QIBM/ProdData/OS400/JT400/lib/JT400Native.jar");

String[] args2 =
{ "-c", "true", "-t", "/path/to/trace", "-o", "/path/to/trace.extension"};

jaCall.setParameters(args2);

if (jaCall.run() != true) {
  // Call Failed
}
```

Prolog 類別

Prolog 類別代表 LAN 線路說明中通訊追蹤的起始 256 位元組區段。「前言」包含追蹤的一般資訊，例如開始與結束時間、收集的位元組數等等。請使用 Prolog 類別，從這個追蹤區段中擷取資訊，隨後您就可透過其他方式來列印、顯示、過濾或處理這一資訊。

Prolog 類別能提供方法，讓您執行各種動作，其中包括：

- 從前言欄位中擷取值，例如追蹤說明、乙太網路類型、資料導向、IP 位址等等。
- 傳回包含前言所有欄位的格式化字串
- 測試前言欄位以找出無效的資料

範例

下列範例將說明，如何利用眾多的 commtrace 類別 (包括 Prolog 類別)，在您的監視器上顯示追蹤資訊：

第 173 頁的『範例：使用 commtrace 類別』

Javadoc 參考文件

有關 Prolog 類別的詳細資訊，請參閱下列 Javadoc 參考文件：

Prolog

Frame 類別

Frame 類別代表 LAN 線路說明的通訊追蹤中，一個記錄或訊框中的所有資料。每一個 Frame 包含三個主要的資料區段，以下列順序出現：

1. 起始的 24 位元組區段，其中包含訊框的一般資訊
2. 訊框的一般資訊 (以 LanHeader 類別代表)
3. 封包資料 (以 IPacket 抽象類別的子類別代表)

使用 Frame 類別，剖析與建立訊框中可列印的資料代表。Frame 類別會維護它所連結之 (使用特定格式的) 清單狀結構中的封包資料。如需訊框中之封包資料可用格式的特定資訊，以及有關訊框結構的一般資訊，請參閱下列項目：

Commtrace 模型

Frame 類別能提供方法， 讓您執行各種動作，其中包括：

- 擷取資料封包
- 擷取訊框的數量、狀態以及類型
- 從訊框傳回特定的資料作爲格式化的字串

您可以使用下列處理程序來存取封包中的資料：

1. 使用 Frame.getPacket() 來擷取封包
2. 透過呼叫 Packet.getHeader() 來存取標頭中的資料
3. 擷取標頭之後，再呼叫 Header.getType() 以尋找類型
4. 使用特定的 Header 子類別，存取該標頭 (Payload) 以及所有其他標頭所關聯的資料

範例

下列範例將說明，如何利用眾多的 commtrace 類別，包括 Format 與 FormatProperties 類別，在您的監視器上顯示追蹤資訊：

第 173 頁的『範例：使用 commtrace 類別』

LanHeader 類別

LanHeader 類別代表訊框中的資料區段，該訊框出現在資訊區段起始的 24 位元組與封包資料之間。這一資料包含有關訊框的一般資訊。

請使用 LanHeader 類別來剖析與列印 LanHeader 中的資訊。 LanHeader 所含的資訊類型包括：

- 能識別此封包中第一個標頭之開始的位元組
- 媒體存取控制 (MAC) 位址
- 記號環位址以及遞送資訊

LanHeader 還提供兩個方法，讓您能夠傳回包含下列項目的格式化「字串」：

- 記號環遞送資料
- 來源檔 MAC 位址、目的地 MAC 位址以及訊框類型

Javadoc 參考文件

有關 LanHeader 類別的詳細資訊，請參閱下列 Javadoc 參考文件：

LanHeader

IPPacket 類別

建立代表特定封包類型的類別時，IPPacket 類別是抽象超類別。IPPacket 的子類別包括：

- ARPPacket
- IP4Packet
- IP6Packet
- UnknownPacket

Packet 類別可讓您擷取封包的類型，並且存取該封包所含的原始資料 (標頭及 Payload)。所有的子類別都使用類似的建構子，並且包含一個額外的方法，該方法可傳回封包內容的可列印版本作爲「字串」。

所有的 Packet 類別建構子都會拿封包資料的位元組陣列作爲引數，但是 ARPPacket 也需要可指定訊框類型的整數。建立 Packet 類別的實例時，會自動建立適當的 Header 物件。

Packet 類別能提供方法，讓您執行各種動作，其中包括：

- 擷取封包的名稱與類型
- 設定封包的類型
- 傳回與封包相關聯的頂層 Header 物件
- 傳回封包資料作爲未格式化的字串
- 從封包傳回特定的資料作爲格式化的字串

Javadoc 參考文件

有關 Packet 類別的詳細資訊，請參閱下列 Javadoc 參考文件：

IPPacket

ARPPacket

IP4Packet

IP6Packet

UnknownPacket

Header 類別

建立代表特定封包標頭類型的類別時， Header 類別是抽象超類別。封包標頭包含了相關聯的資料 (或 Payload)，這些也可以是其他標頭或 Payload。Header 的子類別包括：

- ARPHeader
- ExtHeader
- ICMP4Header
- ICMP6Header
- IP4Header
- IP6Header
- TCPHeader
- UDPHeader
- UnknownHeader

Header 類別可讓您擷取標頭及 Payload 的資料。一個標頭可以封裝其他標頭及其 Payload。

建立 Packet 類別的實例時，會自動建立適當的 Header 物件。Header 類別能提供方法，讓您執行各種動作，其中包括：

- 傳回 Header 的長度、名稱以及類型
- 擷取標頭中作爲位元組陣列的資料
- 擷取封包中的下一個標頭
- 擷取 Payload 作爲位元組陣列、ASCII 字串與十六進位字串
- 傳回標頭資料作爲未格式化的字串
- 從標頭傳回特定的資料作爲格式化的字串

Javadoc 參考文件

有關 Header 類別的詳細資訊，請參閱下列 Javadoc 參考文件：

Header

ARPHeader

ExtHeader

ICMP4Header

ICMP6Header

IP4Header

IP6Header

TCPHeader

UDPHeader

UnknownHeader

範例：使用 commtrace 類別

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// Example using the commtrace classes to print communications trace
// data to a monitor by using a communications trace binary file as
// the source for the data.
//
// Command syntax:
//    java CommTraceExample
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.util.commtrace.*;


public class CommTraceExample {


  public CommTraceExample() {
    // Create a FormatProperties. By default display everything.
    FormatProperties fmtprop = new FormatProperties();

    Format fmt = new Format("/path/to/file");
    // Sets the filtering properties for this format
    fmt.setFilterProperties(fmtprop);
    fmt.formatProlog(); // Format the prolog

    Prolog pro = fmt.getProlog();
    System.out.println(pro.toString());

    // If this is not a valid trace
    if (!pro.invalidData()) {
      Frame rec;

      // Get the records
      while ((rec = fmt.getNextRecord()) != null) {

        // Print out the Frame Number
        System.out.print("Record:" + rec.getRecNum());
        // Print out the time
        System.out.println(" Time:" + rec.getTime());
        // Get this records packet
        IPPacket p = rec.getPacket();
        // Get the first header
        Header h = p.getHeader();

        // If IP6 IPPacket
```

```
        if (p.getType() == IPPacket.IP6) {

          // If IP6 Header
          if (h.getType() == Header.IP6) {

            // Cast to IP6 so we can access methods
            IP6Header ip6 = (IP6Header) h;

            System.out.println(h.getName() + " src:" + ip6.getSrcAddr() + " dst:" + ip6.getDstAddr());
            // Print the header as hex
            System.out.println(ip6.printHexHeader());
            // Print a string representation of the header.
            System.out.println("Complete " + h.getName() + ":\n" + ip6.toString(fmtprop));

            // Get the rest of the headers
            while ((h = h.getNextHeader()) != null) {

              // If it is a TCP header
              if (h.getType() == Header.TCP) {
                // Cast so we can access methods
                TCPHeader tcp = (TCPHeader) h;
                System.out.println(h.getName() +" src:" + tcp.getSrcPort() +" dst:" + tcp.getDstPort());
                System.out.println("Complete " + h.getName() + ":\n" + tcp.toString(fmtprop));

                // If it is a UDP header
              } else if (h.getType() == Header.UDP) {
                // Cast so we can access methods
                UDPHeader udp = (UDPHeader) h;
                System.out.println(h.getName() +" src:" + udp.getSrcPort() + " dst:" + udp.getDstPort());
                System.out.println("Complete " + h.getName() + ":\n" + udp.toString(fmtprop));
              }
            }
          }
        }
      }
    }
  }

    public static void main(String[]args) {
        CommTraceExample e = new CommTraceExample();
    }
}
```

HTML 類別

IBM Toolbox for Java HTML 類別可協助您：

- 設定 HTML 頁的套表及表格
- 對齊文字
- 使用各種 HTML 標記
- 建立「可延伸樣式表語言 (XSL)」格式化物件 (FO) 來源資料
- 改變語言及文字方向
- 建立排序及未排序的清單
- 建立檔案清單及 HTML 階層樹 (以及在其中的元素)
- 新增尚未定義在 HTML 類別中的標示屬性 (如 bgcolor 及 style 屬性)

HTML 類別會執行 HTMLTagElement 介面。每一個類別會產生特定元素類型的 HTML 標記。使用 getTag() 方法可以擷取標示，然後將它們嵌入任何 HTML 文件中。您以 HTML 類別所產生的標示，與 HTML 3.2 規格一致。

HTML 類別可與 Servlet 類別搭配使用，以取得 iSeries 伺服器的資料。然而，如果您提供表格或套表資料，則可以單獨使用該類別。

此外，使用 HTMLDocument 類別也可以讓您輕鬆地建立 HTML 頁或 XSL FO 來源資料。您可以將 XSL FO 資料轉換成「可攜式文件格式 (PDF)」的文件。使用 PDF 格式，可讓您的文件不論在列印時或是以電子方式檢視時，都能維持相同的圖形外觀。

HTML 類別可以讓製作 HTML 套表、表格及其它元素的工作容易些：

- BidiOrdering 類別可讓您改變語言及文字方向。
- DirFilter 類別可讓您確定檔案物件是否爲目錄。
- HTMLAlign 類別可讓您對齊 HTML 輸出的區塊。
- HTMLDocument 類別可讓您更輕鬆地建立 HTML 或 XSL FO 來源資料。
- HTMLFileFilter 類別可讓您確定檔案物件是否爲檔案。
- HTMLForm 類別可協助您比使用「通用閘道介面 SCRIPT」更輕易地製作套表。
- HTMLHead 類別可讓您建立 HTML 頁的標頭標籤。
- HTMLHeading 類別可讓您建立 HTML 頁的標頭標籤。
- HTMLHyperlink 類別可協助您在 HTML 頁中建立鏈接。
- HTMLImage 類別可讓您建立 HTML 頁的影像標籤。
- HTMLList 類別可協助您建立 HTML 頁的清單。
- HTMLMeta 類別可讓您建立 HTML 頁的 meta 標示。
- HTMLParameter 類別指定可供 HTMLServlet 使用的參數。
- HTMLServlet 類別可讓您建立一個伺服器端的併入項目。
- HTMLTable 類別可協助您製作 HTML 頁的表格。
- HTMLText 類別可讓您在您的 HTML 頁中存取字型內容。
- HTMLTree 類別可讓您顯示 HTML 元素的 HTML 階層樹。
- URLEncoder 類別可將要在 URL 字串中使用的區隔字元編碼。
- URLParser 類別可讓您爲 URI、內容及參照剖析 URL 字串。

註: jt400Servlet.jar 檔包括 HTML 和 Servlet 兩種類別。 如果您想要使用在 com.ibm.as400.util.html 套件中的類別，則必須更新 CLASSPATH 並指向 jt400Servlet.jar 檔。

BidiOrdering 類別

BidiOrdering 類別代表一個可改變語言及文字方向的 HTML 標記。HTML <BDO> 字串需要兩個屬性，分別供語言和文字方向使用。

BidiOrdering 類別可讓您：

- 取得及設定語言屬性
- 取得及設定文字的方向

有關使用 <BDO> HTML 標記的資訊，請參閱 W3C  網站。

範例：使用 BidiOrdering

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

下列範例會建立一個 BidiOrdering 物件並設定它的語言及方向：

```
// Create a BidiOrdering object and set the language and direction.
BidiOrdering bdo = new BidiOrdering();
bdo.setDirection(HTMLConstants.RTL);
bdo.setLanguage("AR");


// Create some text.
HTMLText text = new HTMLText("Some Arabic Text.");
text.setBold(true);

// Add the text to the BidiOrdering and get the HTML tag.
bdo.addItem(text);
bdo.getTag();
```

列印陳述式會產生下列標示：

```
<bdo lang="AR" dir="rtl">
  <b>Some Arabic Text.</b>
</bdo>
```

在 HTML 網頁中使用此標籤時，可辨識 <BDO> 標籤的瀏覽器會顯示範例如下：

.txeT cibarA emoS

HTMLAlign 類別

HTMLAlign 類別可讓您對齊 HTML 文件的區段，而非僅對齊個別的項目，例如段落或標頭。

HTMLAlign 類別代表 <DIV> 標籤及其相關的對齊屬性。您可以使用向右、向左、或置中對齊。

您可以使用此類別來執行各種動作，包括：

- 新增或從您要對齊標示的清單移除項目
- 取得及設定對齊
- 取得及設定文字解譯的方向
- 取得及設定輸入元素的語言
- 取得 HTMLAlign 物件的「字串」呈現方式

範例：建立 HTMLAlign 物件

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

下列範例會建立一個未排序的清單，然後建立一個 HTMLAlign 物件來對齊整份清單：

```
// Create an unordered list.
UnorderedList uList = new UnorderedList();
uList.setType(HTMLConstants.DISC);
UnorderedListItem uListItem1 = new UnorderedListItem();
uListItem1.setItemData(new HTMLText("Centered unordered list"));
uList.addListItem(uListItem1);
UnorderedListItem uListItem2 = new UnorderedListItem();
uListItem2.setItemData(new HTMLText("Another item"));
uList.addListItem(uListItem2);
```

```
// Align the list.
HTMLAlign align = new HTMLAlign(uList, HTMLConstants.CENTER);
System.out.println(align);
```

上述範例會產生下列標籤：

```
<div align="center">
<ul type="disc">
  <li>Centered unordered list</li>
  <li>Another item</li>
</ul>
```

在 HTML 網頁中使用此標籤時，會顯示以下畫面：

- 置中對齊的未排序清單
- 其它項目

HTMLDocument 類別

HTMLDocument 類別可讓您更輕鬆地使用現有的 IBM Toolbox for Java HTML 類別，來建立 HTML 頁面或「可攜式文件格式 (PDF)」文件。建立 HTMLDocument 時，您必須指定它包含的是 HTML 標記，還是「可延伸樣式表語言 (XSL)」格式化物件 (FO) 標籤：

- 若要建立 HTML 頁，HTMLDocument 類別可讓您使用比較容易的方法來進行所有必要 HTML 標記的分組。不過，列印出來的 HTML 頁與 Web 瀏覽器所呈現的 HTML 頁，有時會有所不同。
- 若要建立 PDF 文件，HTMLDocument 類別可讓您建立 XSL FO 來源，內含產生 PDF 文件所需的所有資訊。使用 PDF 格式，可讓您的文件不論在列印時或是以電子方式檢視時，都能維持相同的圖形外觀。

若要使用 HTMLDocument，您必須在 CLASSPATH 環境變數中加入 XML 剖析器和 XSLT 處理器。詳細資訊，請參閱下列網頁：

第 10 頁的『Jar 檔』

第 370 頁的『XML 剖析器與 XSLT 處理器』

您可以按照自己想要的方式來處理產生的 HTML 或 XSL 來源資料，例如，顯示 HTML、將 XSL 儲存到檔案中、在 Java 程式的另一個部分中使用串流資料。

有關建立 HTML 頁和 XSL FO 來源資料的資訊，請參閱下列網頁：

- 『使用 HTMLDocument 來建立 HTML 資料』
- 第 178 頁的『使用 HTMLDocument 來建立 XSL FO 資料』
- 第 180 頁的『範例：使用 HTMLDocument』

Javadoc 參考文件

有關 HTMLDocument 類別的資訊，請參閱下列 Javadoc 參考文件：

HTMLDocument

使用 HTMLDocument 來建立 HTML 資料:

HTMLDocument 的作用相當於一個外層，可將建立 HTML 或「可延伸樣式表語言 (XSL)」格式化物件 (FO) 來源資料所需的資訊封裝起來。若要建立 HTML 頁，HTMLDocument 類別可讓您使用比較容易的方法來進行所有必要 HTML 標記的分組。

產生 HTML 來源資料

建立 HTML 來源時，HTMLDocument 會從已建立的 HTML 物件中擷取 HTML 標記。您可以使用 HTMLDocument.getTag() 直接傳送已定義的元素，或針對個別的 HTML 物件使用 getTag()。

HTMLDocument 會依照您在 Java 程式中的定義來產生 HTML 資料，因此請確定結果 HTML 的完整性及正確性。

呼叫 HTMLDocument.getTag() 時，HTMLDocument 物件會執行下列動作：

- 產生開頭的 <HTML> 標籤。並在資料尾端產生結尾的 </HTML> 標籤。
- 將 HTMLHead 和 HTMLMeta 物件轉換成 HTML 標記。
- 緊接著 <HEAD> 標籤之後產生開頭的 <BODY> 標籤。在資料尾端 (結尾的 </HTML> 標籤前面) 會產生結尾的 </BODY> 標籤。

  註: 如果未指定 <HEAD> 標籤， 則 HTMLDocument 會在 <HTML> 標籤後面產生 <BODY> 標籤。
- 依照程式指示，將剩餘的 HTML 物件轉換成 HTML 標記。

註: HTMLDocument 會依照 Java 程式的指示直接傳送 HTML 標記，因此請務必依適當的順序來呼叫標記。

範例：使用 HTMLDocument

下列範例將說明如何使用 HTMLDocument 來產生 HTML 來源資料 (和 XSL FO 來源)：

第 183 頁的『範例：使用 HTMLDocument 來同時產生 HTML 來源和 XSL FO 來源』

Javadoc 參考文件

有關 HTMLDocument 類別的資訊，請參閱下列 Javadoc 參考文件：

HTMLDocument

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

使用 HTMLDocument 來建立 XSL FO 資料:

HTMLDocument 的作用相當於一個外層，可將建立「可延伸樣式表語言 (XSL)」格式化物件 (FO) 或 HTML 來源資料所需的資訊封裝起來。產生的 XSL FO 來源會遵循 XSL FO 格式化模型。此模型使用一些稱爲「區域 (area)」的矩形元素來保留個別的內容元素，這些元素可能是影像、文字、其他 XSL FO 或什麼都不是。下列清單說明四種基本區域：

- 範圍 (Region) 的作用相當於最高層次的儲存區。
- 區塊區域代表區塊層次元素，如段落或清單項目等。

- 行區域代表區塊內的一行文字。
- 內含區域代表一行的某些部分，如單一字元、註腳或數學方程式等。

IBM Toolbox for Java 建立的 XSL FO 標籤會遵循 W3C 建議事項中說明的 XSL 標準。有關 XSL、XSL FOs 和 W3C 建議事項的資訊，請參閱下列內容：

可延伸樣式表語言 (XSL) 1.0 版

產生 XSL FO 來源資料

建立 XSL FO 來源時，HTMLDocument 內容代表可指定頁面大小、方向和邊距的 XSL FO 標籤。此外，HTMLDocument 還會從許多 HTML 類別中擷取該內容元素的對應 XSL FO 標籤。

使用 HTMLDocument 產生 XSL FO 來源之後，您可以使用 XSL 格式製作器 (如 XSLReportWriter 類別)，將內容元素放置於文件的頁面中。

HTMLDocument 會在兩個主要區段中產生 XSL FO 來源資料：

- 第一個區段包含 <fo:root> 和 <fo:layout-master-set> XSL FO 標籤，內含關於頁面高度、頁面寬度和頁面邊距的一般頁面佈置資訊。若要指定佈置資訊的值，請使用 HTMLDocument set 方法來設定相關內容的值。
- 第二個區段包含 XSL FO <fo:page-sequence> 標籤，此標籤含有個別的內容元素。若要指定屬於 HTML 類別案例的個別內容元素， 請從 HTML 物件中擷取對應的 XSL FO 標籤。請務必留意，就內容元素而言，您只能使用具有 getFoTag() 方法的 HTML 類別。

  註: 嘗試從沒有 getFoTag() 方法的 HTML 類別中擷取 XSL FO 標籤，將會造成註解標籤。

針對含有 XSL FO 標籤使用方法的 HTML 類別，如需相關資訊，請參閱下面的 Javadoc 參考文件：

第 180 頁的『可使用 XSL FO 的類別』

建立 HTMLDocument 案例並設定佈置內容之後， 請使用 setUseFO()、getFoTag() 和 getTag() 等方法，從 HTML 物件中擷取 XSL FO 標籤。

- 您可以在 HTMLDocument 或個別的 HTML 物件中使用 setUseFO()。使用 setUseFO() 時，您可以使用 HTMLDocument.getTag() 來擷取 XSL FO 標籤。
- 此外，您還可以在 HTMLDocument 或個別的 HTML 物件中使用 getFoTag() 方法。若您必須從 HTMLDocument 或 HTML 物件中產生 XSL FO 和 HTML 來源時，您可以使用這項替代方法。

範例：使用 HTMLDocument

建立 XSL FO 來源資料後， 您必須將該 XSL FO 資料轉換成使用者可以檢視及列印的形式。下列範例說明如何產生 XSL FO 來源資料 (和 HTML 來源)， 然後使用 XSLReportWriter 和 Context 類別將 XSL FO 來源資料轉換成 PDF 文件：

第 183 頁的『範例：使用 HTMLDocument 來同時產生 HTML 來源和 XSL FO 來源』

第 181 頁的『範例：將 XSL FO 來源資料轉換成 PDF』

Javadoc 參考文件

有關 HTMLDocument 類別的資訊，請參閱下列 Javadoc 參考文件：

HTMLDocument

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

可使用 XSL FO 的類別:

許多 IBM Toolbox for Java HTML 類別具有下列方法，可讓那些類別的實例使用 HTMLDocument：

- getFoTag()
- getTag()
- setUseFO()

針對內含 XSL FO 使用方法的 HTMLDocument 類別和 HTML 類別，如需相關資訊，請參閱下面的 Javadoc 參考文件：

- HTMLDocument
- BidiOrdering
- HTMLAlign
- HTMLHead
- HTMLHeading
- HTMLImage
- HTMLList
- HTMLListItem
- HTMLTable
- HTMLTableCaption
- HTMLTableCell
- HTMLTableHeader
- HTMLTableRow
- HTMLTagElement
- OrderedList
- UnorderedList

範例：使用 HTMLDocument:

下列範例將說明如何使用 HTMLDocument 類別來產生 HTML 和「可延伸樣式表語言 (XSL)」格式化物件 (FO) 來源資料。這兩個範例會將資料直接傳送到...嗎？但您也能處理它？

範例：使用 HTMLDocument 來同時產生 HTML 來源和 XSL FO 來源

下列範例將說明如何同時產生 HTML 來源資料和 XSL FO 來源資料：

第 183 頁的『範例：使用 HTMLDocument 來同時產生 HTML 來源和 XSL FO 來源』

範例：將 XSL FO 來源資料轉換成 PDF

建立 XSL FO 來源資料後， 您必須將該 XSL FO 資料轉換成使用者可以檢視及列印的形式。下面範例將說明，如何使用 XSLReportWriter 和 Context 類別， 將含有 XSL FO 來源資料的檔案轉換成 PDF 文件：

『範例：將 XSL FO 來源資料轉換成 PDF』

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：將 XSL FO 來源資料轉換成 PDF:

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////
//
// Example: Converting XSL FO source to a PDF.
//
// This program uses the IBM Toolbox for Java ReportWriter classes to convert
// XSL FO source data (created by using HTMLDocument) to a PDF.
//
// This example requires the following.jars to be in the classpath.
//
//   composer.jar
//   outputwriters.jar
//   reportwriter.jar
//   x4j400.jar
//   xslparser.jar
//
```

```
// These jars are part of the IBM ToolBox for Java, and reside in directory
// /QIBM/ProdData/HTTP/Public/jt400/lib on your server.
//
// Command syntax:
//    ProcessXslFo FOfilename PDFfilename
//
////////////////////////////////////////////////////////////////////////////


import java.io.FileInputStream;
import java.io.FileOutputStream;

import java.awt.print.Paper;
import java.awt.print.PageFormat;

import org.w3c.dom.Document;

import com.ibm.xsl.composer.framework.Context;

import com.ibm.as400.util.reportwriter.pdfwriter.PDFContext;
import com.ibm.as400.util.reportwriter.processor.XSLReportProcessor;

public class ProcessXslFo
{
    public static void main(String args[])
    {
        if (args.length != 2)
        {
            System.out.println("Usage:  java ProcessXslFo <fo file name> <pdf file name>");
            System.exit(0);
        }

        try
        {
            String inName = args[0];
            String outName = args[1];

            /* Input. File containing XML FO. */
            FileInputStream fin = null;

            /* Output. Which in this example will be PDF. */
            FileOutputStream fout = null;

            try
            {
                fin  = new FileInputStream(inName);
                fout = new FileOutputStream(outName);
            }
            catch (Exception e)
            {
                e.printStackTrace();
                System.exit(0);
            }

            /*
            * Setup Page format.
            */
            Paper paper = new Paper();
            paper.setSize(612, 792);
            paper.setImageableArea(0, 0, 756, 936);

            PageFormat pageFormat = new PageFormat();
            pageFormat.setPaper(paper);

            /*
            * Create a PDF context. Set output file name.
            */
```

```
            PDFContext pdfContext = new PDFContext(fout, pageFormat);

            /*
            * Create XSLReportProcessor instance.
            */
            XSLReportProcessor report = new XSLReportProcessor(pdfContext);

            /*
            * Open XML FO source.
            */
            try
            {
                report.setXSLFOSource(fin);
            }
            catch (Exception e)
            {
                e.printStackTrace();
                System.exit(0);
            }

            /*
            * Process the report.
            */
            try
            {
                report.processReport();
            }
            catch (Exception e)
            {
                e.printStackTrace();
                System.exit(0);
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
            System.exit(0);
        }

        /* exit */
        System.exit(0);
    }
}
```

範例：使用 HTMLDocument 來同時產生 HTML 來源和 XSL FO 來源:

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////
//
// Example: Using the Toolbox HTMLDocument Class
// to generate both HTML and XSL FO source data.
//
// This program uses the HTMLDocument class to
// generate two files: one that has HTML source and
// another than has XSL FO source.
//
// Command syntax:
//    HTMLDocumentExample
//
////////////////////////////////////////////////////////////////////////////


import com.ibm.as400.util.html.*;
import java.*;
import java.io.*;
import java.lang.*;
```

```
import java.beans.PropertyVetoException;

public class HTMLDocumentExample
{
     public static void main(String[] args)
    {
        //Create the HTMLDocument that holds necessary document properties
        HTMLDocument doc = new HTMLDocument();

        //Set page and margin properties.  Numbers are in inches.
        doc.setPageWidth(8.5);
        doc.setPageHeight(11);
        doc.setMarginTop(1);
        doc.setMarginBottom(1);
        doc.setMarginLeft(1);
        doc.setMarginRight(1);

        //Create a header for the page.
        HTMLHead head = new HTMLHead();
        //Set the title for the header
        head.setTitle("This is the page header.");

        //Create several headings
        HTMLHeading h1 = new HTMLHeading(1, "Heading 1");
        HTMLHeading h2 = new HTMLHeading(2, "Heading 2");
        HTMLHeading h3 = new HTMLHeading(3, "Heading 3");
        HTMLHeading h4 = new HTMLHeading(4, "Heading 4");
        HTMLHeading h5 = new HTMLHeading(5, "Heading 5");
        HTMLHeading h6 = new HTMLHeading(6, "Heading 6");

        //Create some text that is printed from right to left.
        //Create BidiOrdering object and set the direction
             BidiOrdering bdo = new BidiOrdering();
             bdo.setDirection(HTMLConstants.RTL);

        //Create some text
        HTMLText text = new HTMLText("This is Arabic text.");
        //Add the text to the bidi-ordering object
             bdo.addItem(text);

                  // Create an UnorderedList.
             UnorderedList uList = new UnorderedList(HTMLConstants.SQUARE);
                  // Create and set the data for UnorderedListItems.
             UnorderedListItem listItem1 = new UnorderedListItem();
             UnorderedListItem listItem2 = new UnorderedListItem();
             listItem1.setItemData(new HTMLText("First item"));
             listItem2.setItemData(new HTMLText("Second item"));
                  // Add the list items to the UnorderedList.
             uList.addListItem(listItem1);
             uList.addListItem(listItem2);

                  // Create an OrderedList.
             OrderedList oList = new OrderedList(HTMLConstants.SMALL_ROMAN);
                  // Create the OrderedListItems.
        OrderedListItem olistItem1 = new OrderedListItem();
        OrderedListItem olistItem2 = new OrderedListItem();
        OrderedListItem olistItem3 = new OrderedListItem();
                  // Set the data in the OrderedListItems.
        olistItem1.setItemData(new HTMLText("First item"));
        olistItem2.setItemData(new HTMLText("Second item"));
        olistItem3.setItemData(new HTMLText("Third item"));
                  // Add the list items to the OrderedList.
        oList.addListItem(olistItem1);
        oList.addListItem(olistItem2);
                  // Add (nest) the unordered list to OrderedListItem2
             oList.addList(uList);
                  // Add another OrderedListItem to the OrderedList
```

```
                // after the nested UnorderedList.
        oList.addListItem(olistItem3);

        // Create a default HTMLTable object.
         HTMLTable table = new HTMLTable();
         try
        {
            // Set the table attributes.
            table.setAlignment(HTMLTable.LEFT);
            table.setBorderWidth(1);

            // Create a default HTMLTableCaption object and set the caption text.
            HTMLTableCaption caption = new HTMLTableCaption();
            caption.setElement("Customer Account Balances - January 1, 2000");

            // Set the caption.
            table.setCaption(caption);

            // Create the table headers and add to the table.
            HTMLTableHeader account_header = new HTMLTableHeader(new HTMLText("ACCOUNT"));
            HTMLTableHeader name_header = new HTMLTableHeader(new HTMLText("NAME"));
            HTMLTableHeader balance_header = new HTMLTableHeader(new HTMLText("BALANCE"));

            table.addColumnHeader(account_header);
            table.addColumnHeader(name_header);
            table.addColumnHeader(balance_header);

            // Add rows to the table.  Each customer record represents a row in the table.
int numCols = 3;
            for (int rowIndex=0; rowIndex< 5; rowIndex++)
            {
                HTMLTableRow row = new HTMLTableRow();
                row.setHorizontalAlignment(HTMLTableRow.CENTER);

                HTMLText account = new HTMLText("000" + rowIndex);
                HTMLText name = new HTMLText("Customer" + rowIndex);
                HTMLText balance = new HTMLText("" + (rowIndex + 1)*200);

                row.addColumn(new HTMLTableCell(account));
                row.addColumn(new HTMLTableCell(name));
                row.addColumn(new HTMLTableCell(balance));

                // Add the row to the table.
                table.addRow(row);
            }
        }
        catch (Exception e)
        {
            System.out.println("Problem creating table");
            System.exit(0);
        }

        //Add the items to the HTMLDocument
        doc.addElement(head);
        doc.addElement(h1);
        doc.addElement(h2);
        doc.addElement(h3);
        doc.addElement(h4);
        doc.addElement(h5);
        doc.addElement(h6);
        doc.addElement(oList);
        doc.addElement(table);
        doc.addElement(bdo);

        //Print the fo tags to a file.
        try
        {
```

```
            FileOutputStream fout = new FileOutputStream("FOFILE.fo");
            PrintStream pout = new PrintStream(fout);
            pout.println(doc.getFOTag());
        }
        catch (Exception e)
        {
            System.out.println("Unable to write fo tags to FOFILE.fo");
        }

        //Print the html tags to a file
        try
        {
            FileOutputStream htmlout = new FileOutputStream("HTMLFILE.html");
            PrintStream phtmlout = new PrintStream(htmlout);
            phtmlout.println(doc.getTag());
        }
        catch (Exception e)
        {
            System.out.println("Unable to write html tags to HTMLFILE.html");
        }

    }
}
```

HTML 套表類別

HTMLForm 類別代表一個 HTML 套表。此類別可讓您：

- 新增元素，如按鈕、超鏈接或 HTML 表到套表中
- 從套表中移除元素
- 設定其它的套表屬性，如使用哪一種方法來傳送套表內容到伺服器、隱藏式參數清單或動作 URL 位址

HTMLForm 物件的建構子會採用 URL 位址。 此位址會被參照爲動作 URL。它是伺服器中，處理套表輸入的應用程式位置。您可以在建構子中指定動作 URL，或使用 setURL() 方法來設定位址。使用不同的設定方法可以設定套表屬性，或使用不同的取得方法擷取它們。

使用 addElement() 可以將任何 HTML 標記元素新增到 HTMLForm 物件，或使用 removeElement() 將它移除。在您的 HTMLForms 中使用下列 HTML 標記元素類別：

- FormInput 類別：代表 HTML 套表的輸入元素
- LayoutFormPanel 類別：代表 HTML 套表的套表元素佈置
- TextAreaFormElement：代表在 HTML 套表中的本文區域元素
- LabelFormElement：代表 HTML 套表元素的標示
- SelectFormElement：代表 HTML 套表的選項輸入類型
- SelectOption：代表 HTML 套表中 SelectFormElement 物件的選項
- RadioFormInputGroup：代表圓鈕輸入物件的群組，可讓使用者從群組中選取一個選項

當然，您也可以新增其它標示元素到套表，包括下列元素：

- HTMLText
- HTMLHyperlink
- HTMLTable

若需更多有關使用 HTMLForm 類別建立套表的資訊，請參閱本範例及結果輸出。

FormInput 類別:

FormInput 類別可讓您：

- 取得及設定輸入元素的名稱
- 取得及設定輸入元素的大小
- 取得及設定輸入元素的起始值

FormInput 類別是由下列清單中的類別加以擴充的。這些類別提供了建立特定類型的套表輸入元素的方法，並可讓您取得及設定輸入元素的各種屬性，或擷取輸入元素的 HTML 標記：

- ButtonFormInput：代表 HTML 套表的按鈕元素
- FileFormInput：代表 HTML 套表的檔案輸入類型
- HiddenFormInput：代表 HTML 套表的隱藏式輸入類型
- ImageFormInput：代表 HTML 套表的影像輸入類型
- ResetFormInput：代表 HTML 套表的重設鈕輸入
- SubmitFormInput：代表 HTML 套表的提出按鈕輸入
- TextFormInput：代表 HTML 套表的單一行文字輸入，您可在其中定義每一行的最大字元數。對於密碼輸入類型，則請您使用 PasswordFormInput，它會擴充 TextFormInput，並代表 HTML 套表的密碼輸入類型
- ToggleFormInput：代表 HTML 套表的輪換輸入類型。使用者可以設定或取出文字標示，並指定是否要勾選或選取輪換。輪換輸入類型可以是兩者之一：
  - RadioFormInput：代表 HTML 套表的圓鈕輸入類型。圓鈕可以放在具有 RadioFormInputGroup 類別的群組中；這會建立一個圓鈕群組，使用者僅可在其中選取其所呈現的選項之一。
  - CheckboxFormInput：代表 HTML 套表的勾選框輸入類型，使用者可在其中選取其所呈現的一個以上的選項，且其中勾選框的起始設定可以是已勾選或未勾選。

ButtonFormInput 類別:

ButtonFormInput 類別代表 HTML 套表的按鈕元素。

下列範例會告訴您如何建立 ButtonFormInput 物件：

```
ButtonFormInput button = new ButtonFormInput("button1", "Press Me", "test()");
System.out.println(button.getTag());
```

此範例會產生下列標籤：

```
<input type="button" name="button1" value="Press Me" onclick="test()" />
```

FileFormInput 類別:

FileFormInput 類別代表 HTML 套表中的檔案輸入類型。

下列程式碼範例會告訴您如何建立新的 FileFormInput 物件

```
FileFormInput file = new FileFormInput("myFile");
    System.out.println(file.getTag());
```

上述程式會建立下列輸出：

```
<input type="file" name="myFile" />
```

HiddenFormInput 類別:

HiddenFormInput 類別代表 HTML 套表中的隱藏式輸入類型。

下列程式碼範例將說明如何建立 HiddenFormInput 物件：

```
HiddenFormInput hidden = new HiddenFormInput("account", "123456");
System.out.println(hidden.getTag()):
```

先前的程式碼會產生下列標籤：

```
<input type="hidden" name="account" value="123456" />
```

在 HTML 網頁中，HiddenInputType 不會顯示。它會將資訊 (此處指帳戶號碼) 傳回伺服器。

ImageFormInput 類別:

ImageFormInput 類別代表 HTML 套表中的影像輸入類型。

您可以使用提供的方法，擷取及更新 ImageFormInput 類別的許多屬性。

- 取得或設定來源
- 取得或設定對齊
- 取得或設定高度
- 取得或設定寬度

範例：建立 ImageFormInput 物件

下面程式範例會告訴您如何建立 ImageFormInput 物件：

```
ImageFormInput image = new ImageFormInput("myPicture", "myPicture.gif");
    image.setAlignment(HTMLConstants.TOP);
    image.setHeight(81);
    image.setWidth(100);
```

上述程式碼範例會產生下列標籤：

```
<input type="image" name="MyPicture" src="myPicture.gif" align="top" height="81" width="100" />
```

ResetFormInput 類別:

ResetFormInput 類別代表 HTML 套表中的重設鈕輸入類型。

下列程式碼範例會告訴您如何建立 ResetFormInput 物件：

```
ResetFormInput reset = new ResetFormInput();
reset.setValue("Reset");
System.out.println(reset.getTag());
```

上述的程式碼範例會產生下列 HTML 標記：

```
<input type="reset" value="Reset" />
```

SubmitFormInput 類別:

SubmitFormInput 類別以 HTML 形式，表示提出按鈕輸入類型。

下列程式碼範例顯示如何建立 SubmitFormInput 物件：

```
SubmitFormInput submit = new SubmitFormInput();
submit.setValue("Send");
System.out.println(submit.getTag());
```

以上的程式碼範例會產生下列輸出：

```
<input type="submit" value="Send" />
```

TextFormInput 類別:

TextFormInput 類別代表 HTML 套表中的單行文字輸入類型。TextFormInput 提供的方法可讓您取得和設定使用者能在文字欄位中輸入的字元數上限。

下列範例會顯示建立新 TextFormInput 物件的方法：

```
TextFormInput text = new TextFormInput("userID");
text.setSize(40);
System.out.println(text.getTag());
```

上述程式碼範例會產生下列標記：

```
<input type="text" name="userID" size="40" />
```

PasswordFormInput 類別:

PasswordFormInput 類別代表 HTML 套表中的密碼輸入欄位類型。

下列程式碼範例會告訴您如何建立新的 PasswordFormInput 物件：

```
PasswordFormInput pwd = new PasswordFormInput("password");
    pwd.setSize(12);
    System.out.println(pwd.getTag());
```

上述程式碼範例會產生下列標記：

```
<input type="password" name="password" size="12" />
```

RadioFormInput 類別:

RadioFormInput 類別代表在 HTML 套表中的圓鈕輸入類型。圓鈕可以在建構時選取並起始設定。

具有相同控制項名稱的一組圓鈕可組成圓鈕群組。RadioFormInputGroup 類別建立圓鈕群組。一次只能選取一個群組內的圓鈕。在建構群組時，也可以選取並起始設定特定的按鈕。

下列程式碼範例會告訴您如何建立一個 RadioFormInput 物件：

```
RadioFormInput radio = new RadioFormInput("age", "twentysomething", "Age 20 - 29", true);
    System.out.println(radio.getTag());
```

上述程式碼範例會產生下列標記：

```
<input type="radio" name="age" value="twentysomething" checked="checked" />
```

CheckboxFormInput 類別:

CheckboxFormInput 類別代表 HTML 套表中的勾選框輸入類型。使用者可以在套表中選取一個以上以勾選框方式呈現的選項。

下列範例會告訴您如何建立新的 CheckboxFormInput 物件：

```
CheckboxFormInput checkbox = new CheckboxFormInput("uscitizen", "yes", "textLabel", true);
System.out.println(checkbox.getTag());
```

上面的程式碼會產生下列輸出：

```
<input type="checkbox" name="uscitizen" value="yes" checked="checked" /> textLabel
```

LayoutFormPanel 類別:

LayoutFormPanel 類別代表 HTML 套表的套表元素佈置。您可以使用 LayoutFormPanel 所提供的方法，在畫面中新增或移除元素，或取得佈置中的元素數目。您可以選擇使用兩種佈置之一：

- 第 190 頁的『GridLayoutFormPanel』：代表 HTML 套表的套表元素格線佈置

- 『LineLayoutFormPanel 類別』：代表 HTML 套表的套表元素行佈置

GridLayoutFormPanel:

GridLayoutFormPanel 類別代表套表元素的格線佈置。您可以指定格線的直欄數，並對 HTML 套表使用此佈置。

下列範例會建立具有兩個直欄的 GridLayoutFormPanel 物件：

```
        // Create a text form input element for the system.
LabelFormElement sysPrompt = new LabelFormElement("System:");
TextFormInput system = new TextFormInput("System");

        // Create a text form input element for the userId.
LabelFormElement userPrompt = new LabelFormElement("User:");
TextFormInput user = new TextFormInput("User");

        // Create a password form input element for the password.
LabelFormElement passwordPrompt = new LabelFormElement("Password:");
PasswordFormInput password = new PasswordFormInput("Password");

        // Create the GridLayoutFormPanel object with two columns and add the form elements.
GridLayoutFormPanel panel = new GridLayoutFormPanel(2);
panel.addElement(sysPrompt);
panel.addElement(system);
panel.addElement(userPrompt);
panel.addElement(user);
panel.addElement(passwordPrompt);
panel.addElement(password);

        // Create the submit button to the form.
SubmitFormInput logonButton = new SubmitFormInput("logon", "Logon");

        // Create HTMLForm object and add the panel to it.
HTMLForm form = new HTMLForm(servletURI);
form.addElement(panel);
form.addElement(logonButton);
```

此範例會產生下列 HTML 程式碼：

```
<form action=servletURI method="get">
<table border="0">
<tr>
<td>System:</td>
<td><input type="text" name="System" /></td>
</tr>
<tr>
<td>User:</td>
<td><input type="text" name="User" /></td>
</tr>
<tr>
<td>Password:</td>
<td><input type="password" name="Password" /></td>
</tr>
</table>
<input type="submit" name="logon" value="Logon" />
</form>
```

LineLayoutFormPanel 類別:

LineLayoutFormPanel 類別代表 HTML 套表的套表元素之行佈置。套表元素會在畫面中以單一列的方式排列。

範例：使用 LineLayoutFormPanel

此範例會建立 LineLayoutFormPanel 物件，並新增兩個套表元素。

```
CheckboxFormInput privacyCheckbox =
   new CheckboxFormInput("confidential", "yes", "Confidential", true);
CheckboxFormInput mailCheckbox =
   new CheckboxFormInput("mailingList", "yes", "Join our mailing list", false);
  LineLayoutFormPanel panel = new LineLayoutFormPanel();
  panel.addElement(privacyCheckbox);
  panel.addElement(mailCheckbox);
  String tag = panel.getTag();
```

上面的程式碼範例會產生下列 HTML 程式碼：

```
<input type="checkbox" name="confidential" value="yes" checked="checked" /> Confidential
<input type="checkbox" name="mailingList" value="yes" /> Join our mailing list <br/>
```

TextAreaFormElement 類別:

TextAreaFormElement 類別代表 HTML 套表中的一個本文區域元素。您可以透過設定列和直欄的數目，來指定本文區域的大小。您可以設定 getRows() 和 getColumns() 方法來決定本文區域元素的大小。

您可以使用 setText() 方法來設定本文區域中的起始文字。您可以使用 getText() 方法來查看已設定的起始文字。

下列範例會顯示建立新 TextAreaFormElement 的方法：

```
TextAreaFormElement textArea = new TextAreaFormElement("foo", 3, 40);
textArea.setText("Default TEXTAREA value goes here");
System.out.println(textArea.getTag());
```

上述程式碼範例會產生下列 HTML 程式碼：

```
<form>
<textarea name="foo" rows="3" cols="40">
Default TEXTAREA value goes here
</textarea>
</form>
```

LabelFormElement 類別:

LabelFormElement 類別代表 HTML 套表元素的標籤。 您可使用 LabelFormElement 類別在 HTML 套表的元素上加上標籤，如本文區域或密碼表單輸入。標籤是您使用 setLabel() 方法設定的一行文字。此文字不會回應使用者輸入，且有了它，可以讓使用者更易於瞭解套表。

範例：使用 LabelFormElement

下面程式會告訴您如何建立 LabelFormElement 物件：

```
LabelFormElement label = new LabelFormElement("Account Balance");
System.out.println(label.getTag());
```

此範例會產生下列輸出：

```
Account Balance
```

SelectFormElement 類別:

SelectFormElement 類別代表 HTML 套表的選取輸入型態。您可以在選取的元素中新增及移除多種選項。

SelectFormElement 提供一些方法，可讓您用來檢視和變更選取元素的屬性：

- 使用 setMultiple() 可以設定使用者是否能選取一個以上的選項
- 使用 getOptionCount() 可以決定選項佈置中有幾項元素
- 使用 setSize() 可以設定選取元素內可見的選項數目，而使用 getSize() 可以決定可見的選項數目。

下列範例會建立一個具有三個選項的 SelectFormElement 物件。名爲 *list* 的 SelectFormElement 物件會以高亮度顯示。新增的前兩個選項可以指定選項文字、名稱以及選取屬性。新增的第三個選項是由 SelectOption 物件定義。

```
SelectFormElement list = new SelectFormElement("list1");
SelectOption option1 = list.addOption("Option1", "opt1");
SelectOption option2 = list.addOption("Option2", "opt2", false);
SelectOption option3 = new SelectOption("Option3", "opt3", true);
list.addOption(option3);
System.out.println(list.getTag());
```

上述的程式碼範例會產生下列 HTML 程式碼：

```
<select name="list1">
<option value="opt1">Option1</option>
<option value="opt2">Option2</option>
<option value="opt3" selected="selected">Option3</option>
</select>
```

SelectOption 類別:

SelectOption 類別代表 HTML SelectFormElement 中的選項。您可以在選取套表中使用選項套表元素。

您可以使用所提供的方法在 SelectOption 中擷取並設定屬性。例如，您可以設定選項預設值是否爲已選取。您也可以設定在提出套表時，它所要使用的輸入值。

下列範例會在一個選取套表中建立三個 SelectOption 物件。 下列的每一個 SelectOption 物件都是以高亮度顯示。他們的名稱是*option1*、*option2* 以及 *option3*。*option3* 物件會在起始時設定成已選取。

```
SelectFormElement list = new SelectFormElement("list1");
SelectOption option1 = list.addOption("Option1", "opt1");
SelectOption option2 = list.addOption("Option2", "opt2", false);
SelectOption option3 = new SelectOption("Option3", "opt3", true);
list.addOption(option3);     System.out.println(list.getTag());
```

上述的程式碼範例會產生下列 HTML 標記：

```
<select name="list1">
<option value="opt1">Option1</option>
<option value="opt2">Option2</option>
<option value="opt3" selected="selected">Option3</option>
</select>
```

RadioFormInputGroup 類別:

RadioFormInputGroup 類別代表一群組的 RadioFormInput 物件。使用者只能從 RadioFormInputGroup 選取一個 RadioFormInput 物件。

RadioFormInputGroup 類別方法可讓您使用圓鈕群組的各種屬性。有了這些方法，您可以：

- 新增圓鈕
- 移除圓鈕
- 取得或設定圓鈕群組的名稱

下面範例會建立圓鈕群組：

```
  // Create some radio buttons.
RadioFormInput radio0 = new RadioFormInput("age", "kid", "0-12", true);
RadioFormInput radio1 = new RadioFormInput("age", "teen", "13-19", false);
RadioFormInput radio2 = new RadioFormInput("age", "twentysomething", "20-29", false);
RadioFormInput radio3 = new RadioFormInput("age", "thirtysomething", "30-39", false);
  // Create a radio button group and add the radio buttons.
```

```
RadioFormInputGroup ageGroup = new RadioFormInputGroup("age");
  ageGroup.add(radio0);
  ageGroup.add(radio1);
  ageGroup.add(radio2);
  ageGroup.add(radio3);
  System.out.println(ageGroup.getTag());
```

上述程式碼範例會產生下列 HTML 程式碼：

```
<input type="radio" name="age" value="kid" checked="checked" /> 0-12
<input type="radio" name="age" value="teen" /> 13-19
<input type="radio" name="age" value="twentysomething" /> 20-29
<input type="radio" name="age" value="thirtysomething" /> 30-39
```

HTMLHead 類別

HTMLHead 類別代表 HTML 標頭標籤。 HTML 網頁的標頭區段裡有開頭和結尾標頭標籤，這些標籤通常含有其他標籤。標頭標籤通常含有標題，也有可能含有 meta 標籤。

HTMLHead 的建構子可讓您建構標頭標籤，包括空的、含有標題標籤的、或是含有標題標籤和 meta 標籤的這幾種。您可以輕鬆地將標題和 meta 標籤新增到空的 HTMLHead 物件上。

HTMLHead 類別的方法包括設定及取得頁面標題和 meta 標籤。請使用 HTMLMeta 類別來定義 meta 標籤的內容。有關 HTMLMeta 類別的資訊，請參閱下一頁：

HTMLMeta

下面程式碼將說明如何建立 HTMLHead 標籤：

```
// Create an empty HTMLHead.
HTMLHead head = new HTMLHead("My Main Page");

// Add the title.
head.setTitle("My main page");

// Define your meta information and add it to HTMLHead.
HTMLMeta meta = new HTMLMeta("Content-Type", "text/html; charset=iso-8859-1");
head.addMetaInformation(meta);
```

這是範例 HTMLHead 標籤的輸出：

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
<title>My main page</title>
</head>
```

Javadoc 參考文件

有關使用 HTMLHead 類別的資訊，請參閱下面的 Javadoc 參考文件：

HTMLHead

HTMLHeading 類別

HTMLHeading 類別代表 HTML 標頭。每個標頭都可以有自己的對齊方式，並具有層次 1 (最大字型、最重要) 到 6 的分別。

HTMLHeading 類別的方法包括：

- 取得及設定標頭的文字

- 取得及設定標頭的層次
- 取得及設定標頭的對齊方式
- 取得及設定文字解譯的方向
- 取得及設定輸入元素的語言
- 取得 HTMLHeading 物件的「字串」呈現方式

範例：建立 HTMLHeading 物件

下列範例會建立三個 HTMLHeading 物件：

```
// Create and display three HTMLHeading objects.
HTMLHeading h1 = new HTMLHeading(1, "Heading", HTMLConstants.LEFT);
HTMLHeading h2 = new HTMLHeading(2, "Subheading", HTMLConstants.CENTER);
HTMLHeading h3 = new HTMLHeading(3, "Item", HTMLConstants.RIGHT);
System.out.print(h1 + "\r\n" + h2 + "\r\n" + h3);
```

前面的範例會產生下列標示：

```
<h1 align="left">Heading</h1>
<h2 align="center">Subheading</h2>
<h3 align="right">Item</h3>
```

HTMLHyperlink 類別

HTMLHyperlink 類別代表 HTML 超鏈接標示。您可使用 HTMLHyperlink 類別，在您的 HTML 頁中建立鏈接。您可以使用此類別取得及設定超鏈接的許多屬性，包括：

- 取得或設定鏈接的「一致資源 ID」
- 取得或設定鏈接的標題
- 取得或設定鏈接的目標框架

HTMLHyperlink 類別可以使用已定義的特性來列印完整的超鏈接，所以您可以在您的 HTML 頁中使用輸出。

下面是 HTMLHyperlink 的範例：

```
// Create an HTML hyperlink to the IBM Toolbox for Java home page.
HTMLHyperlink toolbox =
   new HTMLHyperlink("http://www.ibm.com/as400/toolbox", "IBM Toolbox for Java home page");
toolbox.setTarget(TARGET_BLANK);

// Display the toolbox link tag.
System.out.println(toolbox.toString());
```

上面的程式會產生下列標示：

```
<a href="http://www.ibm.com/as400/toolbox">IBM Toolbox for Java home page</a>
```

HTMLImage 類別

HTMLImage 類別可讓您建立 HTML 頁的影像標示。HTMLImage 類別提供幾種方法讓您取得及設定影像屬性，包括：

- 取得或設定影像的高度
- 取得或設定影像的寬度
- 取得或設定影像的名稱
- 取得或設定影像的替代文字
- 取得或設定影像周圍的水平空間

- 取得或設定影像周圍的垂直空間
- 取得或設定影像的絕對或相對參照
- 擷取 HTMLImage 物件的「字串」呈現方式

下列範例顯示一個建立 HTMLImage 物件的方法：

```
// Create an HTMLImage.
HTMLImage image = new HTMLImage("http://myWebSite/picture.gif", "Alternate text for this graphic");
    image.setHeight(94);
    image.setWidth(105);
    System.out.println(image);
```

列印陳述式會在單行上產生下列標示。覆蓋文字僅供顯示用。

```
<img src="http://myWebSite/picture.gif"
     alt="Alternate text for this graphic"
     height="94" width="105" />
```

HTMLList 類別

HTMLList 可讓您輕鬆地在您的 HTML 頁內建立清單。這些類別提供一些方法，用來取得及設定各種清單屬性和清單內的項目。

尤其是上層類別 HTMLList 提供一個產生壓縮清單的方法，這類清單會儘可能以較小的垂直空間顯示項目。

- HTMLList 的方法包括：
  - 壓縮清單
  - 新增及從清單移除項目
  - 新增及從清單移除清單 (使它儘可能成爲巢狀清單)
- HTMLListItem 的方法包括：
  - 取得及設定項目的內容
  - 取得及設定文字解譯的方向
  - 取得及設定輸入元素的語言

使用 HTMLList 及 HTMLListItem 的次類別來建立您的 HTML 清單：

- OrderedList 及 OrderedListItem
- UnorderedList 及 UnorderedListItem

有關編碼片段，請參閱下列範例：

- 範例：建立排序的清單
- 範例：建立未排序的清單
- 範例：建立巢狀清單

OrderedList 及 OrderedListItem

使用 OrderedList 及 OrderedListItem 類別在您的 HTML 頁中建立排序的清單。

- OrderedList 的方法包括：
  - 取得及設定清單中第一個項目的起始號碼
  - 取得及設定項目號碼的類型 (或樣式)
- OrderedListItem 的方法包括：
  - 取得及設定項目的號碼

– 取得及設定項目號碼的類型 (或樣式)

經由使用 OrderedListItem 中的方法，您可以置換清單中特定項目的編號及類型。

請參閱建立排序的清單範例。

UnorderedList 及 UnorderedListItem

使用 UnorderedList 及 UnorderedListItem 類別在您的 HTML 頁中建立未排序的清單。

- UnorderedList 的方法包括：
  – 取得及設定項目的類型 (或樣式)
- UnorderedListItem 的方法包括：
  – 取得及設定項目的類型 (或樣式)

請參閱建立未排序的清單範例。

範例：使用 HTMLList 類別

下列範例告訴您如何使用 HTMLList 類別來建立排序的清單、未排序的清單及巢狀清單。

範例：建立排序的清單

下列範例會建立一個排序的清單：

```
          // Create an OrderedList.
OrderedList oList = new OrderedList(HTMLConstants.SMALL_ROMAN);
          // Create the OrderedListItems.
OrderedListItem listItem1 = new OrderedListItem();
OrderedListItem listItem2 = new OrderedListItem();
          // Set the data in the OrderedListItems.
listItem1.setItemData(new HTMLText("First item"));
listItem2.setItemData(new HTMLText("Second item"));
          // Add the list items to the OrderedList.
oList.addListItem(listItem1);
oList.addListItem(listItem2);
System.out.println(oList.getTag());
```

前面的範例會產生下列標示：

```
<ol type="i">
<li>First item</li>
<li>Second item</li>
</ol>
```

範例：建立未排序的清單

下列範例會建立一個未排序的清單：

```
          // Create an UnorderedList.
UnorderedList uList = new UnorderedList(HTMLConstants.SQUARE);
          // Create the UnorderedListItems.
UnorderedListItem listItem1 = new UnorderedListItem();
UnorderedListItem listItem2 = new UnorderedListItem();
          // Set the data in the UnorderedListItems.
listItem1.setItemData(new HTMLText("First item"));
listItem2.setItemData(new HTMLText("Second item"));
          // Add the list items to the UnorderedList.
uList.addListItem(listItem1);
uList.addListItem(listItem2);
System.out.println(uList.getTag());
```

前面的範例會產生下列標示：

```
<ul type="square">
<li>First item</li>
<li>Second item</li>
</ul>
```

範例：建立巢狀清單

下列範例會建立一個巢狀清單：

```
          // Create an UnorderedList.
UnorderedList uList = new UnorderedList(HTMLConstants.SQUARE);
          // Create and set the data for UnorderedListItems.
UnorderedListItem listItem1 = new UnorderedListItem();
UnorderedListItem listItem2 = new UnorderedListItem();
listItem1.setItemData(new HTMLText("First item"));
listItem2.setItemData(new HTMLText("Second item"));
          // Add the list items to the UnorderedList.
uList.addListItem(listItem1);
uList.addListItem(listItem2);

          // Create an OrderedList.
OrderedList oList = new OrderedList(HTMLConstants.SMALL_ROMAN);
          // Create the OrderedListItems.
OrderedListItem listItem1 = new OrderedListItem();
OrderedListItem listItem2 = new OrderedListItem();
OrderedListItem listItem3 = new OrderedListItem();
          // Set the data in the OrderedListItems.
listItem1.setItemData(new HTMLText("First item"));
listItem2.setItemData(new HTMLText("Second item"));
listItem3.setItemData(new HTMLText("Third item"));
          // Add the list items to the OrderedList.
oList.addListItem(listItem1);
oList.addListItem(listItem2);
          // Add (nest) the unordered list to OrderedListItem2
oList.addList(uList);
          // Add another OrderedListItem to the OrderedList
          // after the nested UnorderedList.
oList.addListItem(listItem3);
System.out.println(oList.getTag());
```

前面的範例會產生下列標示：

```
<ol type="i">
<li>First item</li>
<li>Second item</li>
<ul type="square">
<li>First item</li>
<li>Second item</li>
</ul>
<li>Third item</li>
</ol>
```

HTMLMeta 類別

HTMLMeta 類別代表在 HTMLHead 標示內使用的 meta 資訊。在 HTML 文件內識別、編索及定義資訊時會使用 META 標示中的屬性。

META 標示的屬性包括：

- NAME - 與 META 標示內容相關的名稱
- CONTENT - 與 NAME 屬性相關的值
- HTTP-EQUIV - 由 HTTP 伺服器針對回覆訊息標頭收集的資訊

- LANG - 語言
- URL - 用來將使用者從現行頁重新導向另一個 URL

例如，若要協助搜尋引擎判定頁面的內容，您可以使用下列 META 標示：

```
<META name="keywords" lang="zh-tw" content="games, cards, bridge">
```

您也可以使用 HTMLMeta 將使用者從某頁重新導向另一頁。

HTMLMeta 類別的方法包括：

- 取得及設定 NAME 屬性
- 取得及設定 CONTENT 屬性
- 取得及設定 HTTP-EQUIV 屬性
- 取得及設定 LANG 屬性
- 取得及設定 URL 屬性

範例：建立 META 標籤

下列範例會建立兩個 META 標示：

```
// Create a META tag to help search engines determine page content.
HTMLMeta meta1 = new HTMLMeta();
meta1.setName("keywords");
meta1.setLang("zh-tw");
meta1.setContent("games, cards, bridge");
// Create a META tag used by caches to determine when to refresh the page.
HTMLMeta meta2 = new HTMLMeta("Expires", "Mon, 01 Jun 2000 12:00:00 GMT");
System.out.print(meta1 + "\r\n" + meta2);
```

前面的範例會產生下列標示：

```
<meta name="keywords" content="games, cards, bridge">
<meta http-equiv="Expires" content="Mon, 01 Jun 2000 12:00:00 GMT">
```

HTMLParameter 類別

HTMLParameter 類別代表可以與 HTMLServlet 類別一起使用的參數。每個參數都有它自己的名稱及值。

HTMLParameter 類別的方法包括：

- 取得並設定參數名稱
- 取得並設定參數值

範例：建立 HTMLParameter 標示

下面範例會建立一個 HTMLParameter 標籤：

```
     // Create an HTMLServletParameter.
HTMLParameter parm = new HTMLParameter ("age", "21");
System.out.println(parm);
```

以上的範例會產生下列標記：

```
<param name="age" value="21">
```

HTMLServlet 類別

HTMLServlet 類別代表一個伺服器端的併入項目。servlet 物件會指定 servlet 的名稱及位置 (後者爲選擇性)。您也可以選擇使用本端系統上的預設位置。

HTMLServlet 類別與 HTMLParameter 類別一起使用，後者會指定可供 servlet 使用的參數。

HTMLServlet 類別的方法包括：

- 新增及從 servlet 標示移除 HTMLParameters
- 取得及設定 servlet 的位置
- 取得及設定 servlet 的名稱
- 取得及設定 servlet 的替代文字

範例：建立 HTMLServlet 標籤

下列爲 HTMLServlet 標示的範例：

```
          // Create an HTMLServlet.
     HTMLServlet servlet = new HTMLServlet("myServlet", "http://server:port/dir");

          // Create a parameter, then add it to the servlet.
     HTMLParameter param = new HTMLParameter("parm1", "value1");
     servlet.addParameter(param);

          // Create and add second parameter
     HTMLParameter param2 = servlet.add("parm2", "value2");

          // Create the alternate text if the Web server does not support the servlet tag.
     servlet.setText("The Web server providing this page does not support the SERVLET tag.")
;
     System.out.println(servlet);
```

前面的範例會產生下列標示：

```
<servlet name="myServlet" codebase="http://server:port/dir">
<param name="parm1" value="value1">
<param name="parm2" value="value2">
     The Web server providing this page does not support the SERVLET tag.
</servlet>
```

HTML Table 類別

HTMLTable 類別可讓您輕易地設定您可使用於 HTML 頁中的表格。此類別提供一些方法以取得及設定表格的各種屬性，包括：

- 取得及設定邊框的寬度
- 取得表格中的列數
- 新增欄或列到表格的末端
- 移除特定欄或列位置上的欄或列

HTMLTable 類別會使用其它的 HTML 類別來讓製作表格的工作更容易些。用來建立表格的其它 HTML 類別是：

- HTMLTableCell：建立表格的資料格
- HTMLTableRow：建立表格的列
- HTMLTableHeader：建立表格的表頭資料格

- HTMLTableCaption：建立表格標題

範例：使用 HTMLTable 類別

下面範例說明使用 HTMLTable 類別的方式：

第 559 頁的『範例：使用 HTMLTable 類別』

HTMLTableCell 類別:

HTMLTableCell 類別會將任何的 HTMLTagElement 物件當作輸入，並以指定的元素建立表格資料格標籤。您可以在建構子上設定元素，或者透過兩種 setElement() 方法的其中之一來設定之。

您可使用 HTMLTableCell 類別中所提供的方法來擷取或更新許多資料格屬性。有些動作您可以使用下列這些方法執行：

- 取得或設定列距
- 取得或設定資料格高度
- 設定資料格資料是否要使用一般 HTML 行分段慣例

下列範例會建立 HTMLTableCell 物件並顯示標示語言：

```
//Create an HTMLHyperlink object.
HTMLHyperlink link = new HTMLHyperlink("http://www.ibm.com",
                     "IBM 首頁");
HTMLTableCell cell = new HTMLTableCell(link);
cell.setHorizontalAlignment(HTMLConstants.CENTER);
System.out.println(cell.getTag());
```

上述 getTag() 方法會提供範例輸出：

<td align=″center″>IBM 首頁 </td>

HTMLTableRow 類別:

HTMLTableRow 類別會在表格中建立一列。此類別提供許多取得和設定列屬性的方法。您可使用這些方法執行下列動作：

- 新增或移除列中的直欄
- 取得直欄資料 (從指定的直欄索引)
- 取得直欄索引 (具有指定資料格的直欄)。
- 取得列中的直欄數
- 設定水平和垂直對齊

以下是 HTMLTableRow 的範例：

```
// Create a row and set the alignment.
HTMLTableRow row = new HTMLTableRow();
row.setHorizontalAlignment(HTMLTableRow.CENTER);

// Create and add the column information to the row.
HTMLText account = new HTMLText(customers_[rowIndex].getAccount());
HTMLText name = new HTMLText(customers_[rowIndex].getName());
HTMLText balance = new HTMLText(customers_[rowIndex].getBalance());

   row.addColumn(new HTMLTableCell(account));
   row.addColumn(new HTMLTableCell(name));
```

```
row.addColumn(new HTMLTableCell(balance));

// Add the row to an HTMLTable object (assume that the table already exists).
table.addRow(row);
```

HTMLTableHeader 類別:

HTMLTableHeader 類別繼承了 HTMLTableCell 類別。它會建立特定的資料格類型，表頭資料格，給您 **<th>** 資料格而不是 **<td>** 資料格。如同 HTMLTableCell 類別，你可呼叫多種方法以更新或擷取標頭資料格的屬性。

以下爲 HTMLTableHeader 的範例：

```
// Create the table headers.
HTMLTableHeader account_header = new HTMLTableHeader(new HTMLText("ACCOUNT"));
HTMLTableHeader name_header = new HTMLTableHeader(new HTMLText("NAME"));
HTMLTableHeader balance_header = new HTMLTableHeader();
HTMLText balance = new HTMLText("BALANCE");
balance_header.setElement(balance);

// Add the table headers to an HTMLTable object (assume that the table already exists).
table.addColumnHeader(account_header);
table.addColumnHeader(name_header);
table.addColumnHeader(balance_header);
```

HTMLTableCaption 類別:

HTMLTableCaption 類別會建立 HTML 表格的標題。此類別提供更新與擷取標題屬性的方法。例如，您可以使用 setAlignment() 方法來指定表格的某部分標題要對齊。以下是 HTMLTableCaption 的範例：

```
// Create a default HTMLTableCaption object and set the caption text.
      HTMLTableCaption caption = new HTMLTableCaption();
caption.setElement("Customer Account Balances - January 1, 2000");

// Add the table caption to an HTMLTable object (assume that the table already exists).
      table.setCaption(caption);
```

HTMLText 類別

HTMLText 類別可讓您存取 HTML 頁的文字內容。利用 HTMLText 類別，您可以取得、設定並檢查許多文字屬性的狀態，包括：

- 取得或設定字型的大小
- 設定粗體屬性開啓 (true) 或關閉 (false)，或確定它是否已開啓
- 設定底線屬性開啓 (true) 或關閉 (false)，或確定它是否已開啓
- 取得或設定文字的水平對齊。

下面範例會告訴您如何建立 HTMLText 物件，以及如何啓用其粗體屬性，並將其字型大小設定爲 5。

```
        HTMLText text = new HTMLText("IBM");
             text.setBold(true);
        text.setSize(5);
   System.out.println(text.getTag());
```

列印陳述式會產生下列標示：

```
     <font size="5"><b>IBM</b></font>
```

在 HTML 網頁中使用此標籤時，會顯示以下畫面：

IBM

HTMLTree 類別

HTMLTree 類別可讓您輕易地設定您可在 HTML 頁中所使用的 HTML 元件階層樹。此類別除了提供可讓您執行下列動作的方法外，還提供一些方法以取得及設定樹的各種屬性：

- 取得及設定 HTTP Servlet 要求
- 對樹新增 HTMLTreeElement
- 從樹移除 HTMLTreeElement

HTMLTree 類別使用其它 HTML 類別，使建立階層樹變得更容易：

- HTMLTreeElement：建立樹元素
- FileTreeElement：建立檔案樹元素
- FileListElement：建立檔案清單元素
- FileListRenderer：提供檔案及目錄清單

範例：使用 HTMLTree 類別

下列範例顯示使用 HTMLTree 類別的不同方式。

- 第 549 頁的『範例：使用 HTMLTree 類別』
- 『範例：建立可遍訪的整合檔案系統樹』

範例：建立可遍訪的整合檔案系統樹:

下列範例是由三個檔案組成，這三個檔案可以一起告訴您如何建立可遍訪的整合檔案系統樹。此範例使用框架以顯示 servlet 中的 HTMLTree 及 FileListElement。

- FileTreeExample.java - 產生 HTML 框架並啓動 servlet
- TreeNav.java - 建置及管理樹
- TreeList.java - 顯示在 TreeNav.java 類別中所做的選擇內容

HTMLTreeElement 類別:

HTMLTreeElement 類別代表 HTMLTree 或其它 HTMLTreeElements 中的階層性元素。

您可使用 HTMLTreeElement 類別中所提供的方法來擷取或更新許多樹元素屬性。有些動作您可以使用下列這些方法執行：

- 取得或設定樹狀元素的可見文字
- 取得或設定已展開及隱藏的圖示之 URL
- 設定 是否要展開樹狀元素

下列範例會建立一個 HTMLTreeElement 物件並顯示其標籤：

```
// Create an HTMLTree.
HTMLTree tree = new HTMLTree();

// Create parent HTMLTreeElement.
HTMLTreeElement parentElement = new HTMLTreeElement();
parentElement.setTextUrl(new HTMLHyperlink("http://myWebPage", "My Web Page"));


// Create HTMLTreeElement Child.
HTMLTreeElement childElement = new HTMLTreeElement();
childElement.setTextUrl(new HTMLHyperlink("http://anotherWebPage", "Another Web Page"));
parentElement.addElement(childElement);
```

```
        // Add the tree element to the tree.
        tree.addElement(parentElement);
        System.out.println(tree.getTag());
```

上述範例中的 getTag() 方法會產生類似如下的 HTML 標記：

```
<table cellpadding="0" cellspacing="3">
<tr>
<td><font color="#0000FF"><u>-</u></font> </td>
<td><font color="#0000FF"><u>My Web Page</u></font></td>
</tr>

<tr>
<td> </td>
<td>
<table cellpadding="0" cellspacing="3">
<tr>
<td><font color="#0000FF"><u>-</u></font> </td>
<td><font color="#0000FF"><u>Another Web Page</u></font> </td>
</tr>
</table></td>
</tr>
</table>
```

FileTreeElement 類別:

FileTreeElement 類別代表 HTMLTree 檢視畫面內的整合檔案系統。

您可使用 HTMLTreeElement 類別中所提供的方法來擷取或更新許多樹元素屬性。您也可以取得及設定 NetServer 共用磁碟機的名稱及路徑。

這些方法能讓您執行的一些動作有：

- 取得或設定展開與隱藏圖示的 URL (繼承的方法)
- 設定是否會展開樹元素 (繼承的方法)
- 取得或設定 NetServer 共用磁碟機的名稱
- 取得或設定 NetServer 共用磁碟機的路徑

範例：使用 FileTreeElement

下列範例會建立 FileTreeElement 物件並顯示標示：

```
        // Create an HTMLTree.
        HTMLTree tree = new HTMLTree();

                // Create a URLParser object.
             URLParser urlParser = new URLParser(httpServletRequest.getRequestURI());

        // Create an AS400 object.
              AS400 system = new AS400(mySystem, myUserId, myPassword);

              // Create an IFSJavaFile object.
              IFSJavaFile root = new IFSJavaFile(system, "/QIBM");

              // Create a DirFilter object and get the directories.
       DirFilter filter = new DirFilter();
              File[] dirList = root.listFiles(filter);

              for (int i=0; i < dirList.length; i++)
       {

           // Create a FileTreeElement.
```

```
            FileTreeElement node = new FileTreeElement(dirList[i]);

            // Set the Icon URL.
            ServletHyperlink sl = new ServletHyperlink(urlParser.getURI());
            sl.setHttpServletResponse(resp);
            element.setIconUrl(sl);

            // Add the FileTreeElement to the tree.
            tree.addElement(element);
        }

        System.out.println(tree.getTag());
```

前述 getTag() 方法提供了範例的輸出。

FileListElement 類別:

FileListElement 類別可讓您建立一個檔案清單元素，它代表整合檔案系統目錄的內容。

您可以取得及設定 NetServer 共用磁碟機的名稱及路徑，使用 FileListElement 物件來代表 NetServer 共用磁碟機的內容。

FileListElement 類別提供可讓您執行下列動作的方法：

- 列出及排序檔案清單的元素
- 取得及設定 HTTP Servlet 要求
- 取得及設定 FileListRenderer
- 取得及設定要用來顯示檔案清單的 HTMLTable
- 取得或設定 NetServer 共用磁碟機的名稱
- 取得或設定 NetServer 共用磁碟機的路徑

您可以與 html 套件中的其它類別一起使用 FileListElement 類別：

- 使用 FileListRenderer，您可以指定您要如何顯示檔案的清單
- 使用 FileTreeElement 類別，您可以建立整合檔案系統檔案或 NetServer 共用檔案的可遍訪清單

FileListElement javadoc 會告訴您如何建立及顯示 FileListElement 物件。

範例：使用 FileListElement 來建立可遍訪的整合檔案系統樹

下列範例告訴您如何與 HTMLTree 類別一起使用 FileListElement 類別 (FileTreeElement 及 HTMLTreeElement) 來建立可遍訪的整合檔案系統樹。範例中也說明了用來設定 NetServer 共用磁碟機路徑的程式碼。

第 202 頁的『範例：建立可遍訪的整合檔案系統樹』

FileListRenderer 類別:

FileListRenderer 類別會提供任何欄位給 FileListElement 中的「檔案」物件 (目錄及檔案)。

FileListRenderer 類別提供可讓您執行下列動作的方法：

- 取得目錄的名稱
- 取得檔案的名稱
- 取得上層目錄的名稱
- 傳回要顯示在 FileListElement 中的列資料

此範例會使用提供器建立 FileListElement 物件：

```
// Create a FileListElement.
FileListElement fileList = new FileListElement(sys, httpservletrequest);

// Set the renderer specific to this servlet, which extends
// FileListRenderer and overrides applicable methods.
fileList.setRenderer(new myFileListRenderer(request));
```

如果您不想使用預設的提供器，您可以擴充 FileListRenderer 並置換新方法或建立新方法。例如，您可能想要確定您防止了傳送特定目錄的名稱，或有特定副檔名的檔案到 FileListElement。藉由擴充此類別並置換適當的方法，您可以傳回這些檔案及目錄的空值，以確定沒有將它們顯示出來。

若要在 FileListElement 內完全自訂列，請使用 getRowData() 方法。在列資料中新增直欄或重新排列直欄，算是使用 getRowData() 來自訂列資料的範例之一。若能接受 FileListRenderer 的預設行爲，您就不需要其他程式設計，因爲 FileListElement 類別會建立預設的 FileListRenderer。

ReportWriter 類別

com.ibm.as400.util.reportwriter 資料包所提供的類別可讓您於使用 iSeries 時，能夠更容易地存取及格式化 XML 來源檔中的資料，或者 Servlet 或 JavaServer Pages(TM) 產生的資料。reportwriter 套件可以很便利地爲三種不同但相關的套件命名：

- com.ibm.as400.util.reportwriter.pclwriter
- com.ibm.as400.util.reportwriter.pdfwriter
- com.ibm.as400.util.reportwriter.processor

這些套件中包含各種類別，可讓您格式化 XML 資料串流以及產生這些格式的報告。請確定您的 CLASSPATH 中有必要的 jar 檔案存在，包括 XML 剖析器與 XSLT 處理器。詳細資訊，請參閱下列網頁：

JAR 檔案

第 370 頁的『XML 剖析器與 XSLT 處理器』

.

Context 類別 (在 pclwriter

- 請使用 PCLContext 搭配上 ReportWriter 類別來產生 Hewlett Packard Printer Control Language (PCL) 格式的報告。
- 請使用 PDFContext 搭配上 ReportWriter 類別來產生 Adobe Portable Document Format (PDF) 格式的報告。

ReportProcessor 類別 (在處理器套件中) 可讓您根據應用程式從 XML 來源資料、Java Servlet 及 JavaServer Pages (JSP) 中所收集的資訊，產生格式化報告。

- 請使用 JSPReportProcessor 類別， 從 Servlets 和 JSP 頁面擷取資料，以產生可用格式 (文字) 的報告。
- 請使用 XSLReportProcessor 類別， 使用 XSL 樣式表處理 XML 資料，以產生可用格式 (環境) 的報告。

Context 類別

「環境」類別支援特定的資料格式，此類別與 OutputQueue 及 SpooledFileOutputStream 類別結合使用之後，可讓 ReportWriter 類別產生該格式的報告，並將這些報告置於排存檔中。

您的應用程式只需建立一個「環境」類別的案例，ReportWriter 類別就會使用它來產生報告。應用程式絕不會直接呼叫這些「環境」類別中的任何方法。PCLContext 及 PDFContext 方法原本就是設計來供 ReportWriter 類別於內部使用。

建構「環境」類別的案例需要 OutputStream (取自於 java.io 套件) 及 PageFormat (取自於 java.awt.print 套件)。下列範例告訴您如何建構並與其它 ReportWriter 類別使用「環境」類別來產生報告：

範例：使用 XSLReportProcessor 含 PCLContext

範例：使用 JSPReportProcessor 與 PDFContext

JSPReportProcessor 類別

JSPReportProcessor 類別可讓您根據 JavaServer Page(TM) (JSP) 或 Java Servlet 的內容，來建立文件或報告。

您可使用此類別從給定的 URL 取得 JSP 或 servlet，並使用其內容產生文件。JSP 或 servlet 必須提供文件資料，包括 XSL 格式化物件。您必須先指定輸出環境 及 JSP 輸入資料來源，才能產生文件的頁面。然後便可將報告資料轉換爲指定的輸出資料串流格式。

JSPReportProcessor 類別可讓您：

- 處理報告
- 將 URL 設定爲範本

下列範例告訴您如何使用 JSPReportProcessor 及 PDFContext 類別來產生報告。這些範例包括 Java 及 JSP 程式碼，您可以使用下列鏈結來檢視它們。您也可以下載 ZIP 檔案， 其中含有 JSPReportProcessor 和 XSLReportProcessor 兩個範例的範例 JSP、XML 和 XSL 來源檔：

- 第 570 頁的『範例：使用 JSPReportProcessor 和 PDFContext』
- 第 572 頁的『範例：JSPReportProcessor 範例 JSP 檔案』

如需 JSP 的相關資訊，請參閱 Java Server Pages technology 網站。

XSLReportProcessor 類別

XSLReportProcessor 類別可讓您使用 XSL 樣式表來轉換及格式化您的 XML 來源資料，進而建立文件或報告。您可以此類別，使用包含 XSL 格式化物件 (FO) 的 XSL 樣式表 (必須符合 XSL 規格)，來建立報告。您接著可使用 Context 類別，將報告資料轉換成指定的輸出資料串流格式。

XSLReportProcessor 類別可讓您：

- 設定 XSL 樣式表
- 設定 XML 資料來源
- 設定 XSL FO 來源
- 處理報告

範例

下列範例將說明如何使用 XSLReportProcessor 及 PCLContext 類別來產生報告。這些範例包括 Java、XML 及 XSL 程式碼，您可以使用下列鏈結來檢視它們。您也可以針對 XSLReportProcessor 和 JSPReportProcessor 這兩個範例，下載 zip 檔，該 zip 檔包含 XML、XSL 和 JSP 原始檔範例：

- 範例：使用 XSLReportProcessor 含 PCLContext

- 範例：XSLReportProcessor 範例 XML 檔
- 範例：XSLReportProcessor 範例 XSL 檔

如需 XML 和 XSL 的詳細資訊，請參閱「資訊中心」的 XML 主題。

資源類別

resource 套件及其類別已棄用。建議您改用 Access 套件。

com.ibm.as400.resource 套件可作爲使用 AS400 各種物件和清單的同屬組織架構。此組織架構能爲所有這類物件和清單提供一致的程式設計介面。

此資源套件包含下列類別：

- Resource - 一種代表 iSeries 資源的物件，例如使用者、印表機、工作、訊息或檔案。資源的具體次類別包括如下：
  - RIFSFile
  - RJavaProgram
  - RJob
  - RPrinter
  - RQueuedMessage
  - RSoftwareResource
  - RUser

  註: 存取套件中的 NetServer 類別也是 Resource 的具體次類別。
- ResourceList - 一種代表 iSeries 資源清單的物件，例如使用者、印表機、工作、訊息或檔案的清單。資源的具體次類別包括如下：
  - RIFSFileList
  - RJobList
  - RJobLog
  - RMessageQueue
  - RPrinterList
  - RUserList
- Presentation - 這是一個物件，可讓您對一般使用者顯示出資源物件、資源清單、屬性、選項和排序用文字格式的相關資訊。

Resource 與 ChangeableResource 類別

resource 套件及其類別已棄用。建議您改用 Acccess 套件。

com.ibm.as400.resource.Resource 及 com.ibm.as400.resource.ChangeableResource 抽象類別代表 iSeries 資源。

Resource

Resource 是提供任何資源屬性之同屬存取的抽象類別。每個屬性都使用屬性 ID 來加以識別，任何給定的 Resource 次類別一般都會把所支援的屬性 ID 記錄在文件中。

Resource 只提供屬性值的讀取權。

IBM Toolbox for Java 提供下列的資源物件：

- RIFSFile - 代表 iSeries 整合檔案系統中的檔案或目錄
- RJavaProgram - 代表 iSeries 中的 Java 程式
- RJob - 代表 iSeries 工作
- RPrinter - 代表 iSeries 印表機
- RQueuedMessage - 代表 iSeries 訊息佇列或工作日誌中的訊息
- RSoftwareResource - 代表 iSeries 中的授權程式
- RUser - 代表 iSeries 使用者

ChangeableResource

ChangeableResource 抽象類別是 Resource 的次類別，新增了 iSeries 資源的屬性值變更能力。屬性變更爲內部快取，直到已確定或已取消爲止。因此可讓您一次變更許多個屬性值。

註: 存取套件中的 NetServer 類別也是 Resource 及 ChangeableResource 的具體次類別。

範例

以下範例說明如何直接使用 Resource 和 ChangeableResource 的具體次類別，以及同屬程式碼如何使用於任何 Resource 或 ChangeableResource 次類別。

- 從 RUser 擷取屬性值，RUser
- 設定 RJob 的屬性值，RJob
- 使用同屬程式碼來存取資源

程式碼範例免責聲明

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

> IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。
>
> IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。
>
> 所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

資源清單

resource 套件及其類別已棄用。建議您改用 Access 套件。

com.ibm.as400.resource.ResourceList 類別代表 iSeries 資源清單。這是提供清單內容的同屬存取的抽象類別。

IBM Toolbox for Java 提供下列的資源清單：

- RIFSFileList - 代表 iSeries 整合檔案系統中的檔案及目錄清單
- RJobList - 代表 iSeries 工作清單
- RJobLog - 代表 iSeries 工作日誌中的訊息清單
- RMessageQueue - 代表 iSeries 訊息佇列中的訊息清單
- RPrinterList - 代表 iSeries 印表機清單
- RUserList - 代表 iSeries 使用者清單

資源清單一律不是開啓就是已關閉。資源清單必須開啓以便存取當中的內容。爲能提供清單內容的立即存取和高效管理記憶體，資源清單大多以遞增方式載入。

資源清單可讓您：

- 開啓清單
- 關閉清單
- 從清單存取特定資源
- 等待特定資源載入
- 等待完整的資源清單載入

您也可以使用選項值來篩選資源清單。每個選項值都使用選項 ID 來加以識別。同樣地，資源清單還可以用排序值來排序。每個排序值都使用排序 ID 來加以識別。ResourceList 的任何給定次類別一般都會把支援的選項 ID 和排序 ID 記錄在文件中。

範例

以下範例說明使用資源清單的各種方式：

- 範例：取得和列印 ResourceList 的內容
- 範例：使用同屬程式碼來存取 ResourceList
- 範例：在 servlet (HTML 表格) 中呈現資源清單

程式碼範例免責聲明

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

> IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。
>
> IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。
>
> 所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

Presentation 類別

resource 套件及其類別已棄用。建議您改用 Access 套件。

每個資源物件、資源清單和 meta 資料物件都有相關的 com.ibm.as400.resource.Presentation 物件可提供已轉換的資訊，例如名稱、完整名稱和圖示。

範例：將資源清單和清單的排序值透過其 Presentation 列印出來

您可使用 Presentation 資訊來把資源物件、資源清單、屬性、選項和排序用文字格式對一般使用者顯示出來。

```
 void printCurrentSort(ResourceList resourceList) throws ResourceException
{
    // Get the presentation for the ResourceList and print its full name.
    Presentation resourceListPresentation = resourceList.getPresentation();
    System.out.println(resourceListPresentation.getFullName());

    // Get the current sort value.
    Object[] sortIDs = resourceList.getSortValue();
```

```
        // Print each sort ID.
        for(int i = 0; i < sortIDs.length; ++i)
        {
            ResourceMetaData sortMetaData = resourceList.getSortMetaData(sortIDs[i]);
            System.out.println("Sorting by " + sortMetaData.getName());
        }
    }
```

Security 類別

使用 IBM Toolbox for Java Security 類別，可以提供與伺服器之間的安全連線、驗證使用者身分，以及在作業系統緒於本端伺服器上執行時，將其與使用者相關聯。包括的安全服務有：

- 使用 Java Secure Socket Extension (JSSE) 的通訊架構，透過將用戶端與伺服器階段作業之間交換的資料加密，以及透過執行伺服器鑑別，來提供安全連線。
- 鑑別服務提供下列功能：
  - 根據 i5/OS 使用者登錄來鑑別使用者身分及密碼。
  - 可將身分指派到目前的 i5/OS 緒。

Secure Sockets Layer

Secure Sockets Layer (SSL) 透過將在用戶端與伺服器階段作業之間交換的資料加密，以及透過執行伺服器鑑別，來提供安全連線。

使用 SSL 對於效能會有負面的影響， 因爲 SSL 連線比沒有加密的連線執行速度慢。當轉送資料的安全性遠比效能重要時 (例如，轉送信用卡或銀行對帳單資訊)， 請使用 SSL 連線。

如需相關資訊，請聯絡 IBM 業務代表。

在 IBM Toolbox for Java 類別及 i5/OS 伺服器之間使用加密

使用 SSL 將 IBM Toolbox for Java 與 i5/OS 伺服器之間的資料加密:

您可使用 SSL 將在 IBM Toolbox for Java 類別與 i5/OS 伺服器之間交換的資料加密。

| 在用戶端上，使用 JSSE 將資料加密。但在伺服器端，您必須使用 i5/OS 數位憑證管理程式來配置 i5/OS 伺
| 服器，才能交換加密的資料。

設置用戶端與伺服器使用 SSL

若要將在 IBM Toolbox for Java 類別與 i5/OS 伺服器之間流通的資料加密，請完成下列作業：

1. 設置您的伺服器交換已加密的資料。
2. 使用 SecureAS400 物件強制 IBM Toolbox for Java 將資料加密。

註: 完成以上兩個步驟只能建立用戶端與伺服器之間的安全路徑。您的應用程式必須使用 SecureAS400 物件來告知 IBM Toolbox for Java 所要加密的資料。唯有流經 SecureAS400 物件的資料才會進行加密。 如果您使用的是 AS400 物件，就不會將資料加密，只會使用一般路徑通往伺服器。

設定 iSeries 伺服器以使用 SSL:

若要設定 iSeries 伺服器使其在 IBM Toolbox for Java 中使用 SSL，請完成下列步驟。

1. 將下列項目安裝到您的 iSeries 伺服器中：

| **註:** 只有執行 V5R4 之前的 i5/OS 版本之伺服器，才需執行此步驟。在 V5R4 及後續版次中，無法再使用
| IBM IBM Cryptographic Access Provider 產品。

- IBM Cryptographic Access Provider for iSeries 128 位元 5722-AC3，可提供伺服器端的加密。

2. 取得並配置伺服器憑證。
3. 將認證套用到 IBM Toolbox for Java 所使用的下列 iSeries 伺服器上：
   - QIBM_OS400_QZBS_SVR_CENTRAL
   - QIBM_OS400_QZBS_SVR_DATABASE
   - QIBM_OS400_QZBS_SVR_DTAQ
   - QIBM_OS400_QZBS_SVR_NETPRT
   - QIBM_OS400_QZBS_SVR_RMTCMD
   - QIBM_OS400_QZBS_SVR_SIGNON
   - QIBM_OS400_QZBS_SVR_FILE
   - QIBM_OS400_QRW_SVR_DDM_DRDA

取得以及配置伺服器憑證

取得以及配置伺服器憑證之前，您必須安裝下列產品：

- IBM HTTP Server for iSeries (5722-DG1) 授權程式
- 基本作業系統選項 34 (數位憑證管理程式)

取得以及配置伺服器憑證所需遵循的程序，是根據您所使用的憑證種類而定：

- 如果您從授信中心 (例如 VeriSign 公司或 RSA Data Security 公司) 取得憑證，請將憑證安裝在 iSeries 上，然後將它套用到主電腦伺服器中。
- 如果您選擇不使用來自授信中心的憑證，您可以建置自己的憑證，以在 iSeries 上使用。使用「數位憑證管理程式」建置憑證：
  1. 在 iSeries 伺服器上建立認證中心。請參閱「資訊中心」主題，作爲您自己的 CA。
  2. 從您建立的認證中心建立系統憑證。
  3. 指定哪些主電腦伺服器要使用您建立的系統憑證。

鑑別服務

類別由 IBM Toolbox for Java 提供，與 i5/OS 提供的安全服務互動。特別是，它也提供了對照 i5/OS 使用者登錄，對使用者身分 (有時候稱爲*主體*) 及密碼進行鑑別的支援。接著便可建立代表已驗證使用者的認證。您可以使用認證來變更現行 i5/OS 緒的身分，利用已驗證使用者的權限及許可權執行工作。實際上，這種 ID 的交換會導致緒的運作如同已驗證的使用者執行登入一般。

註: 建立及交換憑證的服務只有在伺服器的版次是 V5R1M0 或更高時才受支援。

提供的支援概觀

AS400 物件提供特定使用者設定檔以及密碼和伺服器對照的鑑別。您也可以擷取代表系統上已鑑別之使用者設定檔與密碼的 Kerberos 通行證及設定檔記號

註: 使用 Kerberos 通行證前需要先安裝 J2SDK v1.4 並配置「Java 一般安全性服務 (JGSS) 應用程式設計介面」。如需 JGSS 的相關資訊，請參閱 J2SDK, v1.4 Security Documentation 。

若要使用 Kerberos 通行證，只要在 AS400 物件中設定系統名稱 (而非密碼) 即可。 使用者識別是透過 JGSS 組織架構擷取。在 AS400 物件中一次只能設定一種鑑別方式。 設定密碼會清除 Kerberos 通行證或設定檔記號。

若要使用設定檔記號，請使用 getProfileToken() 方法擷取 ProfileTokenCredential 類別的案例。可以將設定檔記號想像成是一個特定伺服器的已驗證使用者設定檔及密碼的代表。設定檔記號的到期基於時間，最多一個小時，但在某些情形下可以重新整理，並提供延伸的生命期限。

註: 如果您使用 ProfileTokenCredential 類別，請務必檢視本頁底下討論設定記號方法的資訊。

下列範例會建立一個系統物件，並使用該物件來產生設定檔記號。然後此範例會使用設定檔記號來建立另一個系統物件，再使用第二個系統物件來連接至指令服務程式：

```
AS400 system = new AS400("mySystemName", "MYUSERID", "MYPASSWORD");
ProfileTokenCredential myPT = system.getProfileToken();
AS400 system2 = new AS400("mySystemName", myPT);
system2.connectService(AS400.COMMAND);
```

設定緒識別

您可以在遠端或本端上下文建立一個認證。一旦建立之後，您可以序列化或分送此認證，一如呼叫應用程式所需。當認證傳遞給相關伺服器上的執行中程序時，可使用認證來修改或交換 i5/OS 緒身分，並以先前已鑑別之使用者身分執行工作。

這種支援的實際應用方式可能是在兩階應用程式中，在第一階的圖形式使用者介面 (如 PC) 執行使用者設定檔及密碼鑑別， 並在第二階 (伺服器) 爲該使用者執行工作。藉由使用 ProfileTokenCredentials，應用程式可以避免直接透過網路傳送使用者 ID 和密碼。接著，此設定檔記號便可分送到第二層的程式，以便執行 *swap()* 並在指派給使用者的 i5/OS 權限及許可權之下操作。

註: 雖然在本質上因爲有限的生命期限，而比傳送使用者設定檔及密碼更具有安全性，但在實際應用以及作相應處理時，設定檔記號仍應視爲敏感資訊。既然記號代表已驗證使用者及密碼，它就有可能被不良的應用程式利用，代表該使用者執行工作。所以，確保該認證是在安全的方式存取是應用程式的責任。

在 ProfileTokenCredential 中設定記號的方法

在 ProfileTokenCredential 類別中設定記號的方法，要求您區別指定密碼的不同方式：

- 透過使用定義的特殊值整數，指定爲特殊值，例如 *NOPWD 或 *NOPWDCHK
- 透過使用代表密碼的「字串」，指定爲使用者設定檔的密碼

註: 在 V5R3 中，IBM Toolbox for Java 不允許不需要區別密碼指定方式的 setToken 方法。

此外，setToken 方法會讓遠端使用者指定密碼特殊值，並且允許長達 128 個字元的使用者設定檔密碼。

若要指定密碼特殊值整數，例如 *NOPWD 或 *NOPWDCHK，請使用下列方法：

- setToken(AS400Principal 主體，int passwordSpecialValue)
- setToken(「字串」名稱， int passwordSpecialValue)

ProfileTokenCredential 類別包括下列用於密碼特殊值整數的靜態常數：

- ProfileTokenCredential.PW_NOPWD: indicates *NOPWD
- ProfileTokenCredential.PW_NOPWDCHK：指出 *NOPWDCHK

若要將使用者設定檔密碼指定爲「字串」，請使用下列方法：

- setTokenExtended(AS400Principal 主體，「字串」密碼)
- setTokenExtended(「字串」名稱，「字串」密碼)

setTokenExended 方法不允許將密碼特殊值字串作爲密碼參數來傳送。例如，這些方法不允許 *NOPWD 密碼字串。

如需詳細資訊，請參閱下列 Javadoc 參考資訊：

ProfileTokenCredential

範例

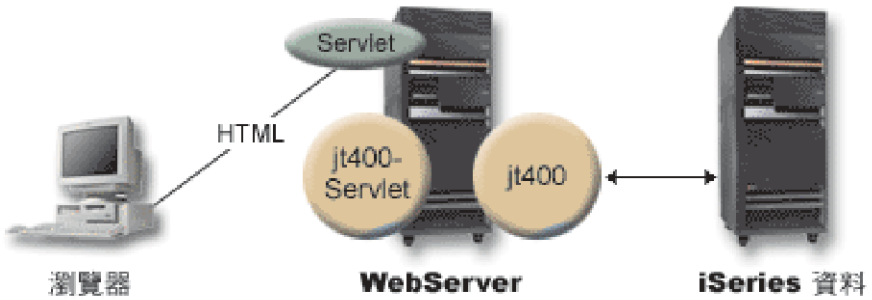
請參閱此程式碼，以取得如何使用設定檔記號認證來交換 i5/OS 緒身分，並以特定使用者的身分執行工作的範例。

Servlet 類別

IBM Toolbox for Java 隨附的 Servlet 類別與位於 Web 伺服器上的存取類別一起使用時，可讓您存取 iSeries 伺服器上的資訊。您可以決定如何使用 Servlet 類別來輔助您自己的 Servlet 專案。

下列圖解將說明 Servlet 類別如何在瀏覽器、Web 伺服器及 iSeries 資料之間運作。瀏覽器會連接到執行 Servlet 的 Web 伺服器。 jt400Servlet.jar 與 jt400.jar files 位於 Web 伺服器上，因爲 Servlet 類別會使用某些存取類別來擷取資料，並以 HTML 類別呈現資料。Web 伺服器則連接到資料所在的 iSeries 伺服器。

圖 1：Servlets 的運作情形



第 214 頁的『圖 1 詳細說明：Servlets 如何運作 (rzahh585.gif)』

IBM Toolbox for Java 內含四種類型的 Servlet 類別：

- Authentication 類別
- RowData 類別
- RowMetaData 類別
- Converter 類別

註：jt400Servlet.jar 檔內含 HTML 及 Servlet 類別。如果您要使用 com.ibm.as400.util.html 以及 com.ibm.as400.util.servlet 套件中的類別，您必須將 CLASSPATH 更新爲指向 jt400Servlet.jar 與 jt400.jar。

有關 Servlets 的一般詳細資訊，請參閱參考資訊區段。

圖 1 詳細說明：Servlets 如何運作 (rzahh585.gif)

位於 IBM Toolbox for Java：Servlet 類別

本圖說明 Servlets 的一般運作方式。

說明

此圖由下列項目所構成：

- 左邊的個人電腦影像，標示著「瀏覽器」，它代表個人電腦上執行的瀏覽器案例。
- 右邊的 iSeries 伺服器影像，標示爲「iSeries 資料」，它代表您要讓 Servlet 存取的資料位置。
- 中間的 iSeries 伺服器影像 (位於其他兩個影像之間)，標示爲 WebServer，代表 Web 伺服器。WebServer 影像上的幾個帶標籤形狀指出位於 WebServer 上的檔案或功能：
  - 標示 Servlet 的綠色橢圓形代表 Servlet 程式碼的位置。
  - 標示 jt400Servlet 的褐色圓形指出 jt400Servlet.jar 檔的位置。
  - 標示 jt400 的褐色圓形指出 jt400.jar 檔的位置。

  註: WebServer 不必 (但可以) 位於 iSeries 伺服器中，它甚至也可以位於「iSeries 資料」影像所指示的同一部伺服器中。
- 連接各個影像的直線。

標示著 HTML 的直線將「瀏覽器」(左邊影像) 連接到 WebServer (中間影像) 上的 Servlet (綠色橢圓形)。此直線標示爲 HTML 的原因爲 Servlets 最常使用 HTML 將資料 '伺服' 到瀏覽器。

WebServer 執行兩個 IBM Toolbox for Java jar 檔 (皮革色圓圈)：jt400Servlet.jar 及 jt400.jar。jt400Servlet.jar 與 jt400.jar 中的類別可讓 WebServer 執行 Servlet，輕易地連接至含有「iSeries 資料」(右邊影像) 的伺服器。兩端都有箭頭，且連接這兩個影像的直線即代表此連接。

Authentication 類別

Servlet 套件中的兩個類別會針對 Servlets：AuthenticationServlet 與 AS400Servlet 執行鑑別。

AuthenticationServlet 類別

AuthenticationServlet 是一個 HttpServlet 施行方式，可執行 Servlets 基本鑑別。AuthenticationServlet 的子類別會置換下列一或多個方法：

- 置換 validateAuthority() 方法以執行鑑別 (必要的)
- 置換 bypassAuthentication() 方法，使得次類別只能鑑別特定的要求
- 置換 postValidation() 方法，可以在鑑別之後處理其它的要求

AuthenticationServlet 類別提供的方法可讓您：

- 起始設定 Servlet
- 取得已鑑別的使用者 ID
- 略過鑑別之後，設定使用者 ID
- 記載異常情況及訊息

AS400Servlet 類別

AS400Servlet 類別是 AuthenticationServlet (代表一個 HTML Servlet) 的摘要次類別。您可以使用連線儲存區來共用連線並管理 Servlet 使用者可擁有的伺服器連線數目。

AS400Servlet 類別提供的方法可讓您：

- 驗證使用者權限 (透過置換 AuthenticationServlet 類別的 validateAuthority() 方法)
- 連接至系統
- 自儲存區取得連線儲存區物件，並將連線儲存區物件傳回儲存區
- 關閉連線儲存區
- 取得並設定 HTML 文件表頭標籤
- 取得並設定 HTML 文件結尾標籤

有關 Servlets 的一般詳細資訊，請參閱參考資訊區段。

RowData 類別

RowData 類別是一個抽象類別，它提供一種說明及存取資料清單的方式。

延伸 RowData 類別的主要有四種類別：

- ListRowData
- RecordListRowData
- ResourceListRowData
- SQLResultSetRowData

RowData 類別可讓您：

- 取得和設定現行位置
- 使用 getObject() 方法取得特定直欄的資料列資料
- 取得列的 meta 資料
- 取得或設定特定直欄的物件內容
- 使用 length() 方法取得清單中的列數。

RowData 位置

有幾種方法可讓您取得並設定清單中的現行位置。下表列出 RowData 類別的 set 和 get 方法。

Set 方法		Get 方法
absolute()	next()	getCurrentPosition()
afterLast()	previous()	isAfterLast()
beforeFirst()	relative()	isBeforeFirst()
first()		isFirst()
last()		isLast()

ListRowData 類別:

ListRowData 類別可讓您執行下列作業：

- 新增及移除結果清單中的列。
- 取得及設定列

- 使用 getMetaData() 方法，取得清單的直欄之相關資訊
- 使用 setMetaData() 方法，設定直欄資訊

ListRowData 類別代表資料清單。透過 IBM Toolbox for Java 第 20 頁的『Access 類別』，ListRowData 可以代表許多種資訊，包括下列各項：

- 整合檔案系統中的目錄
- 工作清單
- 訊息佇列中的訊息清單
- 使用者清單
- 印表機清單
- 排存檔清單

範例

下面範例說明 ListRowData 類別和 HTMLTableConverter 類別的運作方式。此範例說明 Java 程式碼、HTML 程式碼及 HTML 的外觀與操作方式。

第 592 頁的『範例：使用 ListRowData』

RecordListRowData 類別:

RecordListRowData 類別可讓您執行下列工作：

- 將資料列由記錄清單中來回地新增和移除。
- 取得及設定列
- 使用 setRecordFormat 方法設定記錄格式
- 取得記錄格式。

RecordListRowData 類別代表一份記錄清單。可以從伺服器以不同格式取得一筆記錄，包括以下幾種：

- 要寫入伺服器檔案或從伺服器檔案讀取的記錄
- 在資料佇列中的登錄項目
- 來自程式呼叫的參數資料
- 任何需要在伺服器格式與 Java 格式之間作轉換的回傳資料

這個範例顯示 RecordListRowData它會顯示 Java 程式碼、HTML 程式碼及 HTML 的外觀與操作方式。

ResourceListRowData 類別:

ResourceListRowData 類別代表資料的資源清單。請使用 ResourceListRowData 物件來代表任何施行的 ResourceList 介面。

資源清單會格式化成爲一系列的列，各列都含有由直欄數屬性 ID 所決定的有限直欄數。一列以內的每個直欄都含有一個單獨的資料項目。

ResourceListRowData 類別所提供的方法可讓您執行下列動作：

- 取得和設定直欄屬性 ID
- 取得和設定資源清單
- 從清單中擷取列數
- 爲現用列取得直欄資料

- 取得資料物件的內容清單
- 爲清單取得 meta 資料

範例： 在 servlet 中呈現資源清單

程式碼範例免責聲明

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

SQLResultSetRowData 類別:

SQLResultSetRowData 類別將 SQL 結果集呈現成資料清單型態。這個資料是由 SQL 陳述式經由 JDBC 所產生。您可以使用提供的方法，來取得並設定結果集的 meta 資料。

這個範例說明 ListRowData 與 HTMLTableConverter 的工作方式。它會顯示 Java 程式碼、HTML 程式碼及 HTML 的外觀與操作方式。

RowMetaData 類別

RowMetaData 類別定義了一個介面，您可以用它來找出 RowData 物件欄位的相關資訊。

您可以使用 RowMetaData 類別執行下列項目：
- 取得直欄數
- 取得直欄的名稱、類型或大小
- 取得或設定直欄標籤
- 取得直欄資料的精準度或比例
- 判斷直欄資料是否爲文字資料

建置 RowMetaData 類別的主要類別有三種。這些類別除了具有自己的特定功能以外，還提供所有上述的 RowMetaData 功能。
- ListMetaData
- RecordFormatMetaData
- SQLResultSetMetaData

ListMetaData 類別:

ListMetaData 可讓您取得 第 215 頁的『ListRowData 類別』 的相關資訊，以及變更其直欄的設定。 其使用 setColumns() 方法來設定直欄數、清除所有先前的直欄資訊。另外，當您設定建構子的參數時，您也可以略過直欄數。

範例

下面範例說明 ListMetaData、ListRowData 和 HTMLTableConverter 的運作方式。它會顯示 Java 程式碼、HTML 程式碼及 HTML 的外觀與操作方式。

第 592 頁的『範例：使用 ListRowData』

RecordFormatMetaData 類別:

RecordFormatMetaData 可利用 IBM Toolbox for Java RecordFormat 類別。它可讓您在設定建構子的參數時提供記錄格式，或是使用 get 和 set 方法來存取記錄格式。

下列範例會告訴您如何建立 RecordFormatMetaData 物件：

```
// Create a RecordFormatMetaData object from a sequential file's record format.
RecordFormat recordFormat = sequentialFile.getRecordFormat();
RecordFormatMetaData metadata = new RecordFormatMetaData(recordFormat);

// Display the file's column names.
int numberOfColumns = metadata.getColumnCount();
for (int column=0; column < numberOfColumns; column++)
{
    System.out.println(metadata.getColumnName(column));
}
```

SQLResultSetMetaData 類別:

SQLResultSetMetaData 類別會傳回有關 SQLResultSetRowData 物件的直欄資訊。您可以在設定建構子的參數時提供結果集，或是使用 get 和 set 方法來存取結果集的 meta 資料。

下列範例會告訴您如何建立 SQLResultSetMetaData 物件：

```
// Create an SQLResultSetMetaData object from the result set's metadata.
SQLResultSetRowData rowdata = new SQLResultSetRowData(resultSet);
SQLResultSetMetaData sqlMetadata= rowdata.getMetaData();

// Display the column precision for non-text columns.
String name = null;
int numberOfColumns = sqlMetadata.getColumnCount();
for (int column=0; column < numberOfColumns; column++)
{
    name = sqlMetadata.getColumnName(column);
    if (sqlMetadata.isTextData(column))
   {
      System.out.println("Column: " + name + " contains text data.");
   }
   else
   {
      System.out.println("Column: " + name + " has a precision of " + sqlMetadata.getPrecision(column));
   }
}
```

轉換器類別

您可使用轉換器類別，將列資料轉換成格式化的字串陣列。結果會以 HTML 套表呈現，並可供 HTML 網頁使用。下列類別會替您處理轉換：

- StringConverter
- HTMLFormConverter
- HTMLTableConverter

StringConverter 類別:

StringConverter 類別爲一抽象類別，它代表的是一列資料字串的轉換器。它提供了 convert() 方法，可用來轉換列資料。這會傳回呈現該列資料的字串陣列。

HTMLFormConverter 類別:

HTMLFormConverter 類別會擴充 StringConverter，方法是經由提供額外的轉換方式，此方式稱爲 convertToForms()。此方法會將列資料轉換成單一列 HTML 表格的陣列。您可以使用這些表格標示，在瀏覽器中顯示格式化的資訊。

您可以使用不同的取得及設定方法，檢視或變更套表的屬性以裁製 HTML 套表的外觀。例如，您可以設定的部分屬性包括：

- 對齊
- 資料格間距
- 標頭超連結
- 寬度

範例：使用 HTMLFormConverter

下列範例說明如何使用 HTMLFormConverter。(您可以在執行中的 Web 伺服器上編譯及執行此範例。)

使用 HTMLFormConverter

HTMLTableConverter 類別:

HTMLTableConverter 類別藉由提供 convertToTables() 方法，來延伸 StringConverter。此方法將資料列轉換成 HTML 表格陣列，以便讓 Servlet 用來在瀏覽器中顯示清單。

您可以使用 getTable() 及 setTable() 方法，選擇轉換期間要使用的預設表格。您可以在 HTML 表格物件中設定表格的標頭，或使用標頭資訊的 meta 資料，方法爲設定 setUseMetaData() 爲 TRUE。

setMaximumTableSize() 方法可讓您限制單一表格的列數。如果資料列超出指定的表格大小，則轉換器會在輸出陣列中，產生另一個 HTML 表格物件。此會一直繼續到所有資料列都被轉換完畢。

範例

下面範例說明如何使用 HTMLTableConverter 類別：

- 範例：使用 ListRowData
- 範例：使用 RecordListRowData
- 範例：使用 SQLResultSetRowData
- 範例：顯示 Servlet 中的 ResourceList

程式碼範例免責聲明

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

Utility 類別

公用程式類別可以幫助您執行特定的作業。

IBM Toolbox for Java 提供下列公用程式：

- AS400ToolboxJarMaker：產生載入速度較快的 IBM Toolbox for Java JAR 檔，其方法是從較大的 JAR 檔建立一個較小的 JAR 檔，或選擇解壓縮一個 JAR 檔來存取個別的內容檔案。
- CommandHelpRetriever：擷取並產生 i5/OS 控制語言 (CL) 指令的說明文字。
- CommandPrompter：提示輸入給定指令的參數。CommandPrompter 提供的功能類似於 iSeries CL 指令提示 (按 F4)，且與「管理中心」指令提示相同。
- RunJavaApplication 及 VRunJavaApplication：可讓您在 iSeries 伺服器上，從指令行提示執行 Java 程式。
- JPing：可讓您查詢伺服器，找出作用中的服務。您也可以指定是否要連通測試 (ping) SSL 埠。

用戶端安裝與更新類別

在伺服器的整合檔案系統中，IBM Toolbox for Java 類別可在其位置上被參考到。

因爲暫時修訂程式 (PTF) 套用至此位置，所以在伺服器中直接存取這些類別的 Java 程式會自動接收這些更新。但是，存取伺服器的類別並非每次都可成功，尤其是在下列情況下：

- 如果使用低速通訊鏈接連接伺服器與用戶端，則從伺服器載入類別的效能可能很低。
- 如果 Java 應用程式使用 CLASSPATH 環境變數來存取用戶端檔案系統上的類別，則您需要 iSeries Access for Windows，方可將檔案系統呼叫重新導向至伺服器。iSeries Access for Windows 可能無法位於用戶端。

在這些情況中，在用戶端中安裝這些類別是較佳的解決方案。

AS400ToolboxJarMaker

雖然 JAR 檔案格式的設計目的是在加快 Java 程式檔的下載速度，但 AS400ToolboxJarMaker 可透過本身的能力，從較大的 JAR 檔建立一個較小的 JAR檔，進而產生一個載入速度更快的 IBM Toolbox for Java JAR 檔。

AS400ToolboxJarMaker 類別

同時，AS400ToolboxJarMaker 類別可以替您解壓縮 JAR 檔，以便能存取個別的內容檔以進行基本使用。

AS400ToolboxJarMaker 的彈性

所有 AS400ToolboxJarMaker 功能均是透過 JarMaker 類別及 AS400ToolboxJarMaker 次類別來執行：

- 同屬 JarMaker 工具可在任何 JAR
- AS400ToolboxJarMaker 可自訂及擴充 JarMaker 功能，以便更易於使用 IBM Toolbox for Java JAR 檔。

依據您的需求，您可以從自己的 Java 程式或指令行呼叫 AS400ToolboxJarMaker 方法。請使用下列語法從指令行呼叫 AS400ToolboxJarMaker：

```
java utilities.JarMaker [options]
```

其中

- 選項 = 一或多個可用選項

欲取得可在指令行提示上執行的選項的完整集，請參閱下列：

- JarMaker 基礎類別的選項
- AS00ToolboxJarMaker 子類別的擴充選項

使用 AS400ToolboxJarMaker

您可透過幾種方法來搭配使用 AS400ToolboxJarMaker 與 JAR 檔：

- 解壓縮 JAR 檔內的一個檔案
- 將大的 JAR 檔分割成較小的 JAR 檔
- 排除任何應用程式不需要執行的 IBM Toolbox for Java 檔案

解壓縮 JAR 檔

假設您僅想要解壓縮 JAR 檔案內的一個檔案集。AS400ToolboxJarMaker 可讓您將檔案展開到下列其中一項：

- 現行目錄 (extract(jarFile))
- 另一個目錄 (extract(jarFile, outputDirectory))

例如，透過下列程式碼，您會從 jt400.jar 取出 AS400.class 及其所有的相依類別：

```
java utilities.AS400ToolboxJarMaker -source jt400.jar
        -extract outputDir
        -requiredFile com/ibm/as400/access/AS400.class
```

將一個 JAR 檔分割成幾個較小的 JAR 檔

假設您想要依照您對 JAR 檔大小上限的偏好，將一個大型的 JAR 檔分割成較小的 JAR 檔。AS400ToolboxJarMaker 會相對地提供您 split(jarFile, splitSize) 功能。

在下列程式碼中，jt400.jar 會分割成一組較小的 JAR 檔，每個檔案都在 300KB 以下：

```
java utilities.AS400ToolboxJarMaker -split 300
```

從 JAR 檔中移除未使用的檔案

使用 AS400ToolboxJarMaker，您可以只選取應用程式執行所需的 IBM Toolbox for Java 元件、語言及 CCSID，以排除應用程式不需要的 IBM Toolbox for Java 檔案。AS400ToolboxJarMaker 也提供您一個選項，讓您可以併入或排除您所選取元件相關的 JavaBean 檔。

例如，下列指令所建立的 JAR 檔，只會包含讓IBM Toolbox for Java 的 CommandCall 及 ProgramCall 元件生效所需的 IBM Toolbox for Java 類別：

```
java utilities.AS400ToolboxJarMaker -component CommandCall,ProgramCall
```

此外，若不須在 Unicode 與雙位元組字集 (DBCS) 轉換表之間轉換字串，您可以使用 -ccsid 選項略過不需要的轉換表，來建立 400KB 位元組的較小 JAR 檔：

```
java utilities.AS400ToolboxJarMaker -component CommandCall,ProgramCall -ccsid 61952
```

註: 程式呼叫類別不包含轉換類別。在併入程式呼叫類別時，程式所使用的轉換類別也必須經由 -ccsid 選項明確地併入。

在 IBM Toolbox for Java 中受支援的元件:

下列表格列出呼叫 AS400ToolboxJarMaker 工具時可指定的元件 ID。

- 「元件」直欄會列出元件的一般名稱。
- 「關鍵字」直欄列出您在使用 `-component` 選項標籤時應該指定的關鍵字。
- 「常數」直欄會列出在 setComponents() 和 getComponents() 中應指定的整數值。

元件	關鍵字	常數
伺服器物件	AS400	AS400ToolboxJarMaker.AS400
指令呼叫	CommandCall	AS400ToolboxJarMaker.COMMAND_CALL
連線儲存區	ConnectionPool	AS400ToolboxJarMaker.CONNECTION_POOL
資料區	DataArea	AS400ToolboxJarMaker.DATA_AREA
資料說明與轉換	DataDescription	AS400ToolboxJarMaker.DATA_DESCRIPTION
資料佇列	DataQueue	AS400ToolboxJarMaker.DATA_QUEUE
數位憑證	DigitalCertificate	AS400ToolboxJarMaker.DIGITAL_CERTIFICATE
FTP	FTP	AS400ToolboxJarMaker.FTP
整合檔案系統	IntegratedFileSystem	AS400ToolboxJarMaker.INTEGRATED_FILE_SYSTEM
JAAS	JAAS	AS400ToolboxJarMaker.JAAS
Java 應用程式呼叫	JavaApplicationCall	AS400ToolboxJarMaker.JAVA_APPLICATION_CALL
JDBC	JDBC	AS400ToolboxJarMaker.JDBC
工具及工具佇列	Job	AS400ToolboxJarMaker.JOB
訊息及訊息佇列	Message	AS400ToolboxJarMaker.MESSAGE
數值資料類型	NumericDataTypes	AS400ToolboxJarMaker.NUMERIC_DATA_TYPES
NetServer	NetServer	AS400ToolboxJarMaker.NETSERVER
網路列印	Print	AS400ToolboxJarMaker.PRINT
程式呼叫	ProgramCall	AS400ToolboxJarMaker.PROGRAM_CALL
記錄層次存取	RecordLevelAccess	AS400ToolboxJarMaker.RECORD_LEVEL_ACCESS
安全伺服器	SecureAS400	AS400ToolboxJarMaker.SECURE_AS400
服務程式呼叫	ServiceProgramCall	AS400ToolboxJarMaker.SERVICE_PROGRAM_CALL
系統狀態	SystemStatus	AS400ToolboxJarMaker.SYSTEM_STATUS
系統值	SystemValue	AS400ToolboxJarMaker.SYSTEM_VALUE
追蹤及記載	Trace	AS400ToolboxJarMaker.TRACE
使用者和群組	User	AS400ToolboxJarMaker.USER
使用者空間	UserSpace	AS400ToolboxJarMaker.USER_SPACE
目視伺服器物件	AS400Visual	AS400ToolboxJarMaker.AS400_VISUAL
目視指令呼叫	CommandCallVisual	AS400ToolboxJarMaker.COMMAND_CALL_VISUAL
目視資料佇列	DataQueueVisual	AS400ToolboxJarMaker.DATA_QUEUE_VISUAL
目視整合檔案系統	IntegratedFileSystemVisual	AS400ToolboxJarMaker.INTEGRATED_FILE_SYSTEM_VISUAL
視覺化 Java 應用程式呼叫	JavaApplicationCallVisual	AS400ToolboxJarMaker.JAVA_APPLICATION_CALL_VISUAL
目視 JDBC	JDBCVisual	AS400ToolboxJarMaker.JDBC_VISUAL
目視工作及工作佇列	JobVisual	AS400ToolboxJarMaker.JOB_VISUAL
目視訊息及訊息佇列	MessageVisual	AS400ToolboxJarMaker.MESSAGE_VISUAL

元件	關鍵字	常數
目視網路列印	PrintVisual	AS400ToolboxJarMaker.PRINT_VISUAL
目視程式呼叫	ProgramCallVisual	AS400ToolboxJarMaker.PROGRAM_CALL_VISUAL
目視記錄層次存取	RecordLevelAccessVisual	AS400ToolboxJarMaker.RECORD_LEVEL_ACCESS_VISUAL
目視使用者與群組	UserVisual	AS400ToolboxJarMaker.USER_VISUAL

IBM Toolbox for Java 支援的 CCSID 及編碼值:

IBM Toolbox for Java 附有一組根據 CCSID 命名的轉換表。當對 iSeries 伺服器的往來傳送資料進行轉換時，IBM Toolbox for Java 類別 (例如 CharConverter) 會在內部使用這些表格。例如，CCSID 1027 的轉換表位於 com/ibm/as400/access/ConvTable1027.class 檔案中。 IBM Toolbox for Java JAR 檔含有下列 CCSID 的轉換表；其他編碼則透過 JDK 的使用來支援。在執行期間不再使用伺服器上的中央伺服器來下載表格。任何轉換表或 JDK 編碼找不到的指定 CCSID 將會導致異常發生。其中有些表格對於內含在您 JDK 中的表格而言可能是多餘的。IBM Toolbox for Java 目前支援下列 122 個不同的 i5/OS CCSID。

如需 CCSID 的相關資訊 (包括 iSeries 伺服器所能辨識的完整 CCSID 清單)，請參閱全球化。

在 IBM Toolbox for Java 中受支援的 CCSID

CCSID	格式	說明
37	單位元組 EBCDIC	美國與其它地區
273	單位元組 EBCDIC	澳洲、德國
277	單位元組 EBCDIC	丹麥、挪威
278	單位元組 EBCDIC	芬蘭、瑞典
280	單位元組 EBCDIC	義大利
284	單位元組 EBCDIC	西班牙、拉丁美洲
285	單位元組 EBCDIC	英國
290	單位元組 EBCDIC	日文片假名 (限單位元組)
297	單位元組 EBCDIC	法國
300	雙位元組 EBCDIC	日文圖形 (16684 的子集)
367	ASCII/ISO/Windows	ASCII (ANSI X3.4 標準)
420	單位元組 EBCDIC (雙向)	阿拉伯文 EBCDIC ST4
423	單位元組 EBCDIC	希臘文 (用於相容性；請參閱 875)
424	單位元組 EBCDIC (雙向)	希伯來文 EBCDIC ST4
437	ASCII/ISO/Windows	ASCII (USA PC 資料)
500	單位元組 EBCDIC	拉丁語-1 (MNCS)
720	ASCII/ISO/Windows	阿拉伯文 (MS-DOS)
737	ASCII/ISO/Windows	希臘文 (MS-DOS)
775	ASCII/ISO/Windows	波羅的海語系 (MS-DOS)
813	ASCII/ISO/Windows	ISO 8859-7 (希臘文/拉丁文)
819	ASCII/ISO/Windows	ISO 8859-1 (拉丁文-1)
833	單位元組 EBCDIC	韓文 (限單位元組)
834	雙位元組 EBCDIC	韓文圖形 (4930 的子集)
835	雙位元組 EBCDIC	繁體中文圖形

CCSID	格式	說明
836	單位元組 EBCDIC	簡體中文 (限單位元組)
837	雙位元組 EBCDIC	簡體中文圖形
838	單位元組 EBCDIC	泰文
850	ASCII/ISO/Windows	拉丁文-1
851	ASCII/ISO/Windows	希臘文
852	ASCII/ISO/Windows	拉丁文-2
855	ASCII/ISO/Windows	斯拉夫文
857	ASCII/ISO/Windows	土耳其文
860	ASCII/ISO/Windows	葡萄牙文
861	ASCII/ISO/Windows	冰島
862	ASCII/ISO/Windows (雙向)	希伯來文 ASCII ST4
863	ASCII/ISO/Windows	加拿大
864	ASCII/ISO/Windows (雙向)	阿拉伯文 ASCII ST5
865	ASCII/ISO/Windows	丹麥/挪威
866	ASCII/ISO/Windows	斯拉夫文/俄文
869	ASCII/ISO/Windows	希臘文
870	單位元組 EBCDIC	拉丁文-2
871	單位元組 EBCDIC	冰島
874	ASCII/ISO/Windows	泰文 (9066 的子集)
875	單位元組 EBCDIC	希臘文
878	ASCII/ISO/Windows	俄文
880	單位元組 EBCDIC	多種語言斯拉夫文 (用於相容性；請參閱 1025)
912	ASCII/ISO/Windows	ISO 8859-2 (拉丁文-2)
914	ASCII/ISO/Windows	ISO 8859-4 (拉丁文-4)
915	ASCII/ISO/Windows	ISO 8859-5 (八位元的斯拉夫文)
916	ASCII/ISO/Windows (雙向)	ISO 8859-8 (希伯來文) ST5
920	ASCII/ISO/Windows	ISO 8859-9 (拉丁文-5)
921	ASCII/ISO/Windows	ISO 8859-13 (八位元的波羅的海語系)
922	ASCII/ISO/Windows	愛沙尼亞 ISO-8
923	ASCII/ISO/Windows	ISO 8859-15 (拉丁文-9)
930	混合位元組 EBCDIC	日文 (5026 的子集)
933	混合位元組 EBCDIC	韓文 (1364 的子集)
935	混合位元組 EBCDIC	簡體中文 (1388 的子集)
937	混合位元組 EBCDIC	繁體中文
939	混合位元組 EBCDIC	日文 (5035 的子集)
1025	單位元組 EBCDIC	斯拉夫文
1026	單位元組 EBCDIC	土耳其文
1027	單位元組 EBCDIC	日文拉丁文 (限單位元組)
1046	ASCII/ISO/Windows (雙向)	Windows 阿拉伯文 ST5
1089	ASCII/ISO/Windows (雙向)	ISO 8859-6 (阿拉伯文) ST5

CCSID	格式	說明
1112	單位元組 EBCDIC	波羅的海語系多種語言
1122	單位元組 EBCDIC	愛沙尼亞
1123	單位元組 EBCDIC	烏克蘭
1125	ASCII/ISO/Windows	烏克蘭
1129	ASCII/ISO/Windows	越南文
1130	單位元組 EBCDIC	越南文
1131	ASCII/ISO/Windows	白俄羅斯
1132	單位元組 EBCDIC	寮國
1140	單位元組 EBCDIC	美國與其訰地區 (歐元支援)
1141	單位元組 EBCDIC	澳洲、德國 (歐元支援)
1142	單位元組 EBCDIC	丹麥、挪威 (歐元支援)
1143	單位元組 EBCDIC	芬蘭、瑞典 (歐元支援)
1144	單位元組 EBCDIC	義大利 (歐元支援)
1145	單位元組 EBCDIC	西班牙、拉丁美洲 (歐元支援)
1146	單位元組 EBCDIC	英國 (歐元支援)
1147	單位元組 EBCDIC	法國 (歐元支援)
1148	單位元組 EBCDIC	拉丁-1 (MNCS) (歐元支援)
1149	單位元組 EBCDIC	冰島 (歐元支援)
1200	Unicode	Unicode UCS-2 (小位元組排序法)
1250	ASCII/ISO/Windows	Windows 拉丁文-2
1251	ASCII/ISO/Windows	Windows 斯拉夫文
1252	ASCII/ISO/Windows	Windows 拉丁文-1
1253	ASCII/ISO/Windows	Windows 希臘文
1254	ASCII/ISO/Windows	Windows 土耳其文
1255	ASCII/ISO/Windows (雙向)	Windows 希伯來文 ST5
1256	ASCII/ISO/Windows (雙向)	Windows 阿拉伯文 ST5
1257	ASCII/ISO/Windows	Windows 波羅的海語系
1258	ASCII/ISO/Windows	Windows 越南文
1364	混合位元組 EBCDIC	日文
1388	混合位元組 EBCDIC	簡體中文
1399	混合位元組 EBCDIC	日文 (在 V4R5 或更新的版本中)
4396	雙位元組 EBCDIC	日文 (300 的子集)
4930	雙位元組 EBCDIC	韓文
4931	雙位元組 EBCDIC	繁體中文 (835 的子集)
4933	雙位元組 EBCDIC	簡體中文 GBK 圖形
4948	ASCII/ISO/Windows	拉丁文-2 (852 的子集)
4951	ASCII/ISO/Windows	斯拉夫文 (855 的子集)
5026	混合位元組 EBCDIC	日文
5035	混合位元組 EBCDIC	日文
5123	單位元組 EBCDIC	日文 (限單位元組、歐元支援)
5351	ASCII/ISO/Windows (雙向)	Windows 希伯來文 (歐洲字元支援) ST5

CCSID	格式	說明
8492	雙位元組 EBCDIC	日文 (300 的子集)
8612	單位元組 EBCDIC	阿拉伯文 EBCDIC ST5
9026	雙位元組 EBCDIC	韓文 (834 的子集)
9029	雙位元組 EBCDIC	簡體中文 (4933 的子集)
9066	ASCII/ISO/Windows	泰文 (延伸 SBCS)
12588	雙位元組 EBCDIC	日文 (300 的子集)
13122	雙位元組 EBCDIC	韓文 (834 的子集)
16684	雙位元組 EBCDIC	日文 (可於 V4R5 中取得)
17218	雙位元組 EBCDIC	韓文 (834 的子集)
12708	單位元組 EBCDIC	阿拉伯文 EBCDIC ST7
13488	Unicode	Unicode UCS-2 (大位元組排序法)
28709	單位元組 EBCDIC	繁體中文 (限單位元組)
61952	Unicode	iSeries Unicode (主要用於整合檔案系統)
62211	單位元組 EBCDIC	希伯來文 EBCDIC ST5
62224	單位元組 EBCDIC	阿拉伯文 EBCDIC ST6
62235	單位元組 EBCDIC	希伯來文 EBCDIC ST6
62245	單位元組 EBCDIC	希伯來文 EBCDIC ST10

CommandHelpRetriever 類別

CommandHelpRetriever 類別會擷取 i5/OS 控制語言 (CL) 指令的說明文字，並以 HTML 或「使用者介面管理程式 (UIM)」這兩種格式之一來產生該文字。您可以從指令行來執行 CommandHelpRetriever，或者將該功能內嵌在您的 Java 程式中。

若要使用 CommandHelpRetriever，您的伺服器就必須執行 i5/OS V5R1 或更新版本，而且在 CLASSPATH 環境變數中具有 XML 剖析器與 XSL 處理器。如需詳細資訊，請參閱下列網頁：

第 370 頁的『XML 剖析器與 XSLT 處理器』

此外，「建立指令文件 (GENCMDDOC)」CL 指令也會使用 CommandHelpRetriever 類別。因此，您只要使用 GENCMDDOC 指令，就能夠利用 CommandHelpRetriever 類別所提供的功能。如需詳細資訊，請參閱下列網頁：

建立指令文件 (GENCMDDOC)

從指令行執行 CommandHelpRetriever

您可以將 CommandHelpRetriever 類別作爲獨立式指令行程式來執行。您必須傳送下列最基本的參數，才能從指令行執行 CommandHelpRetriever：

- iSeries 伺服器上包含 CL 指令的檔案庫。系統指令位於 QSYS 檔案庫。
- CL 指令。

您也可以傳送選用性參數給包含 iSeries 伺服器、使用者 ID、密碼以及產生檔案所在位置的 CommandHelpRetriever。

如需詳細資訊，請參閱 Javadoc reference documentation for CommandHelpRetriever。

範例：從指令行使用 CommandHelpRetriever

下列範例會在現行目錄中產生一個名爲 CRTLIB.html 的 HTML 檔。

註: 範例指令出現在兩行上，這純粹是爲了方便顯示。請您在單一行中輸入指令。

```
java com.ibm.as400.util.CommandHelpRetriever -library QSYS -command CRTLIB
     -system MySystem -userid MyUserID -password MyPassword
```

在您的程式中內含 CommandHelpRetriever 類別

您也可以在 Java 應用程式中使用 CommandHelpRetriever 類別，以顯示指定 CL 指令的說明文件。當您建立 CommandHelpRetriever 物件之後， 您可以使用 generateHTML 以及 generateUIM 方法，以其中一種格式來產生說明文件。

如果您使用 generateHTML()，您就可以在指令的畫面群組中顯示所產生的 HTML，或者也可以指定不同的畫面群組。

下列範例會建立 CommandHelpRetriever 物件，並產生 String 物件，來代表 CRTLIB 指令的 HTML 與 UIM 說明文件。

```
CommandHelpRetriever helpGenerator = new CommandHelpRetriever();
AS400 system = new AS400("MySystem", "MyUserID", "MyPassword");
Command crtlibCommand = new Command(system, "/QSYS.LIB/CRTLIB.CMD");
String html = helpGenerator.generateHTML(crtlibCommand);
String uim = helpGenerator.generateUIM(crtlibCommand);
```

Javadoc 參考文件

如需有關 CommandHelpRetriever 類別的詳細資訊，請參閱下列 Javadoc 參考文件：

CommandHelpRetriever

CommandPrompter 類別

CommandPrompter 類別會提示給定指令的參數。CommandPrompter 提供類似 iSeries CL 指令提示 (按 F4) 以及與管理中心指令提示相同的功能。

若要使用 CommandPrompter，伺服器必須執行 i5/OS V4R4 或更新版本。如需相關資訊，請參閱 iSeries Navigator Information APARs，並檢視 Required Fixes for Graphical Command Prompter Support。

使用 CommandPrompter 亦需要您在 CLASSPATH 中具有下列 jar 檔：

- jt400.jar
- jui400.jar
- util400.jar
- jhall.jar

在 CLASSPATH 中也必須有 XML Parser。如需 XML 剖析器適當使用方式的相關資訊，請參閱下一頁：

第 370 頁的『XML 剖析器與 XSLT 處理器』

IBM Toolbox for Java 包括所有 JAR 檔，除了 jhall.jar 以外。如需 IBM Toolbox for Java JAR 檔的相關資訊，請參閱 JAR 檔。關於下載 jhall.jar 的詳細資訊，請參閱 Sun JavaHelp(TM) 網站  。

若要建構 CommandPrompter 物件，您必須傳送啓動提示器的上層框架參數、提示指令所在的 AS400 物件、以及指令字串給它。指令字串可以是指令名稱、完整指令字串、或部分指令名稱，例如 `crt*`。

CommandPrompter 顯示器是一個使用者在返回上層框架之前，必須先關閉的模組對話框。CommandPrompter 會處理提示期間出現的任何錯誤。 有關說明 CommandPrompter 使用方式的程式設計範例，請參閱下一頁：

第 657 頁的『範例：使用 CommandPrompter』

RunJavaApplication

RunJavaApplication 及 VRunJavaApplication 類別是公用程式，可用來在 iSeries JVM 上執行 Java 程式。與必須從您的 Java 程式呼叫的 JavaApplicationCall 及 VJavaApplicationCall 類別不同，RunJavaApplication 及 VRunJavaApplication 是完整的程式。

RunJavaApplication 類別爲指令行公用程式。它可讓您設定 Java 程式的環境 (例如，CLASSPATH 及內容)。您可以先指定 Java 程式的名稱及其參數，然後啓動程式。一旦啓動，您可以傳送「輸入」至 Java 程式，它可以透過標準輸入裝置接收。Java 程式將輸出寫入標準輸出及標準錯誤。

VRunJavaApplication 公用程式具有相同的功能。不同的是 VJavaApplicationCall 使用圖形式使用者介面時，JavaApplicationCall 爲指令行介面。

JPing 類別

JPing 類別爲一指令行公用程式，可讓您查詢伺服器以找出正在執行的服務爲何，及提供服務的埠有哪些。若要從 Java 應用程式內部進行伺服器的查詢，請使用 AS400JPing 類別。

如需在 Java 應用程式內使用 JPing 的相關資訊，請參閱 JPing javadoc。

使用下列語法，由指令行呼叫 JPing：

```
java utilities.JPing System [options]
```

其中：

- System = 您要查詢的 iSeries 伺服器
- [options] = 一或多個可用選項

選項

您可使用下列選項之一或其中數個項目。如果選項有縮寫，縮寫會列於括弧內。

-help (-h or -?)
顯示說明本文。

-service *i5/OS_Service* **(-s** *i5/OS_Service***)**
指定一項要進行連通測試 (ping) 的特定服務。預設動作爲對所有服務進行連通測試 (ping)。您可使用此選項來指定下列其中一項服務：
as-file、as-netprt、as-rmtcmd、as-dtaq、as-database、as-ddm、as-central 及 as-signon。

-ssl 指定是否要對 ssl 埠進行連通測試 (ping)。 預設動作爲不要對 ssl 埠進行連通測試 (ping)。

-timeout (-t)
指定逾時期間 (以毫秒爲單位)。預設設定爲 20000，或 20 秒。

範例：在指令行使用 JPing

例如，使用下列指令對 as-dtaq 服務 (包括 ssl 埠) 進行連通測試 (ping)， 逾時時間爲 5 秒：

```
java utilities.JPing myServer -s as-dtaq -ssl -t 5000
```

Vaccess 類別

Vaccess 套件及其類別已棄用。建議您改爲結合使用 Access 套件及 Java Swing。

IBM Toolbox for Java 在 Vaccess 套件中提供一組圖形式使用者介面 (GUI) 類別。這些類別會使用存取類別來擷取資料，並呈現資料供使用者查看。

使用 IBM Toolbox for Java Vaccess 類別的 Java 程式，必須具備 Java 2 Platform 標準版 (J2SE) 隨附的 Swing 套件。如需 Swing 的相關資訊，請參閱 Sun Java Foundation Classes 網站。

如需 IBM Toolbox for Java GUI 類別、Access 類別及 Java Swing 之間關係的相關資訊，請參閱 Vaccess 類別圖解。

您可以使用 AS400 窗格類別，來顯示 iSeries 資料。

API 可用來存取下列 iSeries 資源及其工具：

- 指令呼叫
- 資料佇列
- 錯誤事件*
- 整合檔案系統
- JavaApplicationCall
- JDBC
- 工作*
- 訊息*
- 許可權
- 列印* 包括已排存的檔案檢視器
- ProgramCall 及 ProgramParameter
- 記錄層次存取
- 資源清單
- 系統狀態
- 系統值
- 使用者和群組

註: AS400 窗格會與其他 Vaccess 類別搭配使用 (請參閱前述以星號標示的項目)，來顯示 iSeries 資源並允許其操作。

以 IBM Toolbox for Java 圖形式使用者介面元件進行程式設計時，請使用「錯誤事件」類別，向使用者報告及處理錯誤事件。

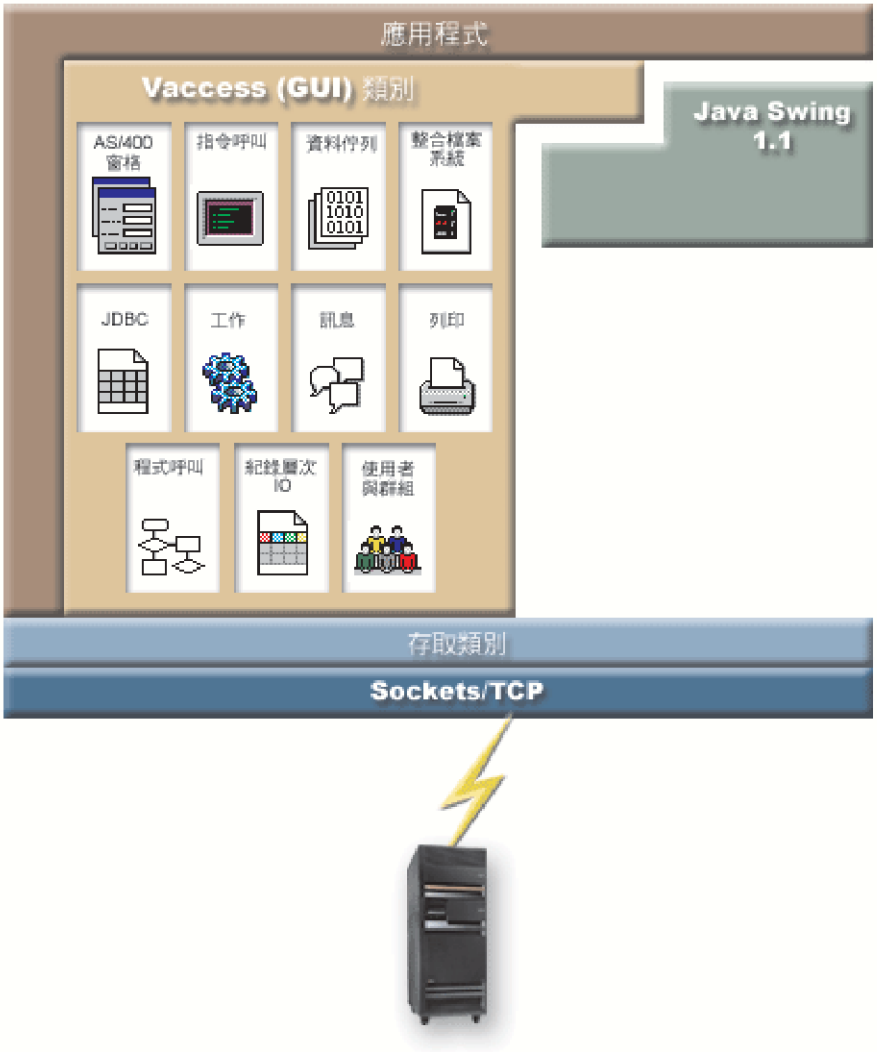
如需存取 iSeries 資料的相關資訊，請參閱 Access 類別。

Vaccess 類別

Vaccess 套件及其類別已棄用。建議您改爲結合使用 Access 套件及 Java Swing。

IBM Toolbox for Java 在 Vaccess 套件中提供圖形式使用者介面 (GUI) 類別，以擷取及顯示伺服器資料，並在某些情況下操作該伺服器資料。這些類別使用 Java Swing 1.1 組織架構。圖 1 顯示這些類別之間的關係：

圖 1：Vaccess 類別



『圖 1 的長說明：Vaccess 類別 (rzahh508.gif)』

圖 1 的長說明：Vaccess 類別 (rzahh508.gif):

位於 IBM Toolbox for Java：Vaccess 套件圖解

本圖說明 Vaccess 套件、存取套件及 Java Swing 類別中類別之間的關係。

說明

此圖由下列項目所構成：

- 最頂端的影像實際上是矩形左上側的棕色粗邊框，標示爲「應用程式」，代表 Java 應用程式。Java 應用程式劃分出來的矩形包含下列形狀不規則的影像，可像拼圖般合起來：

- 皮革色多邊形，代表 IBM Toolbox for Java Vaccess (GUI) 類別。 這個形狀中包含更小的影像，代表 Vaccess 類別中所含的函數。
- 綠色多邊形，代表 Java Swing 類別
- 這些影像下方有道標示爲「Access 類別」的淺藍色長列，代表 IBM Toolbox for Java 存取套件中的類別。
- 淺藍長列的下方有形狀相同的深藍長列，標示爲 Sockets/TCP。
- 底端的影像是 iSeries 伺服器的圖片。
- 一道閃電，代表從 Java 應用程式到伺服器的 Socket 連線，從 Socket/TCP (深藍色長列) 向下延伸之後連接到伺服器 (iSeries 伺服器的影像)。

Java 應用程式 (棕色粗邊框所劃分的區域) 含有 Vaccess 類別 (皮革色多邊形) 及 Java Swing 類別 (綠色多邊形)。Vaccess 類別使得 Java 應用程式能夠存取伺服器中的下列資料及功能 (皮革色多邊形中的小影像)：

AS400Panes、CommandCall、DataQueues、整合檔案系統、JDBC、工作、訊息、列印、ProgramCall、記錄層級的輸出/入、和使用者與群組

Java 應用程式會使用 IBM Toolbox for Java Access 類別 (淺藍色長列)，來建立一或多個 Socket 連線 (深藍色長列)。Socket 連線使 Java 應用程式能夠與伺服器 (底端的 iSeries 伺服器影像) 通訊 (閃電)。

AS400Panes

AS400Panes 是 vaccess 套件的元件，它們會在 GUI 中呈現及容許操作一個或多個伺服器資源。每一個伺服器資源的行爲會隨著資源類型而有所不同。

所有窗格都會延伸 Java Component 類別。因此，它們可新增到任何「AWT 頁框」、「視窗」或「容器」中。

下列是可用的 AS400Panes：

- AS400DetailsPane 會在表格中呈現伺服器資源清單，其中的每一列則顯示單一資源的各種詳細資訊。表格容許選擇一個或多個資源。
- AS400ExplorerPane 結合了 AS400TreePane
- AS400JDBCDataSourcePane 會呈現 AS400JDBCDataSource 物件的內容值。
- AS400ListPane 會呈現伺服器資源清單，並容許選擇一或多個資源。
- AS400TreePane 會以樹狀階層結構呈現伺服器資源，並容許選擇一或多個資源。

伺服器資源

在圖形使用者介面中，伺服器資源是以圖示及文字來表示。伺服器資源是透過階層關係來定義，因此資源可能有一個上層，以及零或多個子項。這些是預先定義的關係，而且可以用來指定哪些資源要顯示在 AS400Pane 中。例如，VJobList 是零或多個 VJobs 的上層，而且這種階層關係在 AS/400Pane 中會以圖形方式來表示。

IBM Toolbox for Java 可讓您存取下列伺服器資源：

- VIFSDirectory 代表整合檔案系統中的目錄
- VJob 和 VJobList 分別代表工作或工作清單
- VMessageList 和 VMessageQueue 分別代表從 CommandCall 或 ProgramCall 或訊息佇列傳回的訊息清單
- VPrinter、VPrinters 和 VPrinterOutput 分別代表印表機、印表機清單或排存檔的清單
- VUserList 代表使用者清單

所有資源都是 VNode 介面的實作。

設定根

若要指定哪些伺服器資源會呈現在 AS400Pane 中，請使用建構子或 setRoot() 方法設定根。根物件會依據窗格來定義不同方式使用的最上層物件：

- AS400ListPane 在其清單中呈現根的所有子項
- AS400DetailsPane 在其表格中呈現根的所有子項
- AS400TreePane 使用根作爲其樹狀結構的根
- AS400ExplorerPane 使用根作爲其樹狀結構的根

窗格與根物件可做任意組合。

下列範例會建立一個 AS400DetailsPane，來呈現系統中所定義的使用者清單：

```
                        // Create the server resource
                        // representing a list of users.
                        // Assume that "system" is an AS400
                        // object created and initialized
                        // elsewhere.
     VUserList userList = new VUserList (system);

                        // Create the AS400DetailsPane object
                        // and set its root to be the user
                        // list.

AS400DetailsPane detailsPane = new AS400DetailsPane ();

detailsPane.setRoot (userList);

                        // Add the details pane to a frame.
                        // Assume that "frame" is a JFrame
                        // created elsewhere.
      frame.getContentPane ().add (detailsPane);
```

載入內容

當建立 AS400Pane 物件與伺服器資源物件時，會將它們起始設定爲預設狀態。在建立期間，不會載入構成窗格內容的相關資訊。

若要載入內容，應用程式必須以明確方式呼叫 load() 方法。在大多數情況下，這會起始與伺服器的通訊，來收集相關資訊。因爲有時會花上一段時間來收集此資訊，所以應用程式可以精確地控制收集時間。例如，您可以：

- 將窗格新增到頁框之前先載入內容。頁框不會出現，直到載入所有資訊爲止。
- 將窗格新增到頁框且顯示該頁框之後再載入內容。頁框會出現，但它不會含有許多資訊。「等待游標」會出現且在載入後該資訊就會填入。

下列範例會在將明細窗格的內容新增到頁框之前，先載入該內容：

```
                        // Load the contents of the details
                        // pane. Assume that the detailsPane
                        // was created and initialized
                        // elsewhere.

detailsPane.load ();

                        // Add the details pane to a frame.
                        // Assume that "frame" is a JFrame
                        // created elsewhere.
     frame.getContentPane ().add (detailsPane);
```

動作與特性窗格

在執行時間，使用者可以選取任何伺服器資源中的蹦現功能表。蹦現功能表會呈現可供資源使用的相關動作的清單。當使用者選取蹦現功能表的一個動作時，即會執行該動作。每一資源均有一個不同的已定義動作。

在某些情況下，蹦現功能表也會呈現一個容許使用者檢視特性窗格的項目。特性窗格會顯示關於資源的不同明細，且可能容許使用者變更那些明細。

該應用程式可以在窗格中使用 setAllowActions() 方法，來控制是否可以使用動作與內容窗格。

模型

AS400Panes 是使用模型-概略表-控制器參照範例來實作，其中的資料與使用者介面被分成不同的類別。AS400Panes 可整合 IBM Toolbox for Java 模型與 Java GUI 元件。這些模型會管理伺服器資源，而 vaccess 元件則會以圖形方式來顯示它們及處理使用者互動。

AS400Panes 會提供足夠的功能來符合大部分的需求。不過，如果應用程式需要更進一步控制 JFC 元件，則應用程式可以直接存取伺服器模型，並提供與不同的 vaccess 元件的自訂整合。

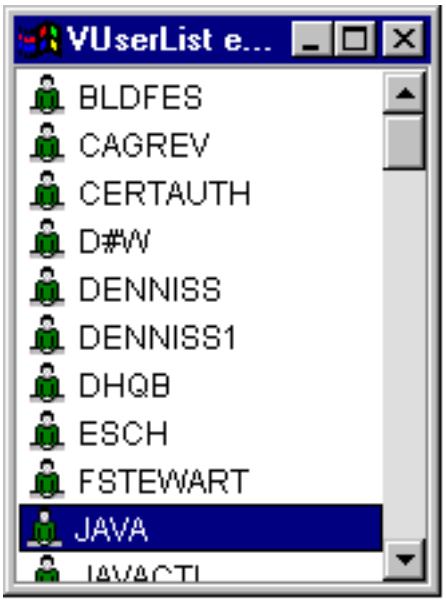
下列是可用的模型：

- AS400ListModel 會將 JFC ListModel 介面實作爲伺服器資源的清單。這可與 JFC JList 物件搭配使用。
- AS400DetailsModel 將 JFC TableModel 介面實作爲伺服器資源的表格，其中的每一列皆包含單一資源的各種相關明細。這可與 JFC JTable 物件搭配使用。
- AS400TreeModel 會將 JFC TreeModel 介面實作爲伺服器資源的樹狀階層。這可與 JFC JTree物件搭配使用。

範例

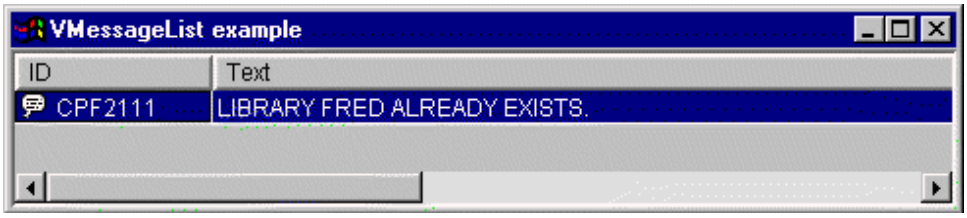
- 使用 AS400ListPane 與 VUserList 物件，在系統上呈現使用者清單。圖 1 顯示完成的產品：

  圖 1：搭配使用 AS400ListPane 和 VUserList 物件



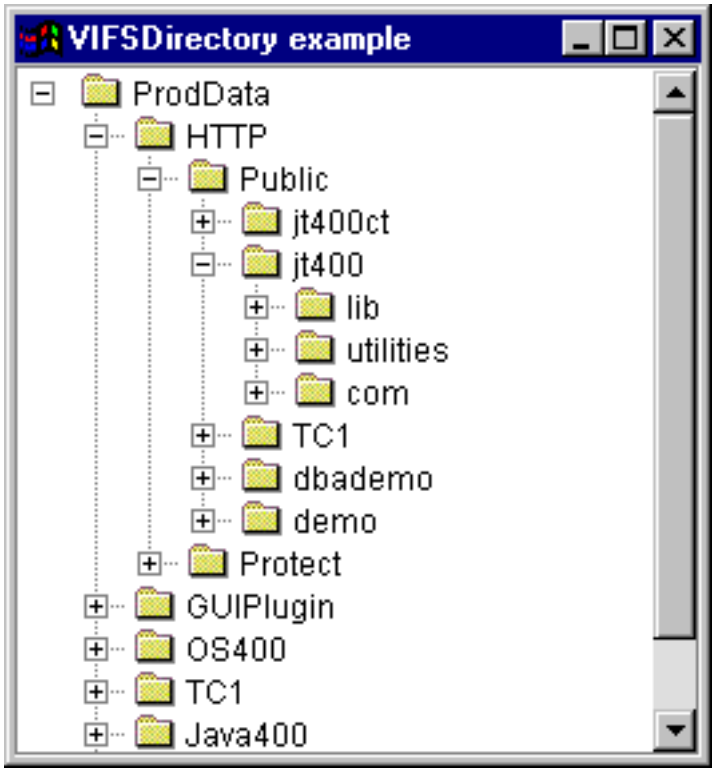
- 使用 VMessageList 物件的 AS400DetailsPane，來呈現指令呼叫所產生的訊息清單。圖 2 顯示完成的產品：

  圖 2：搭配使用 AS400DetailsPane 和 VMessageList 物件



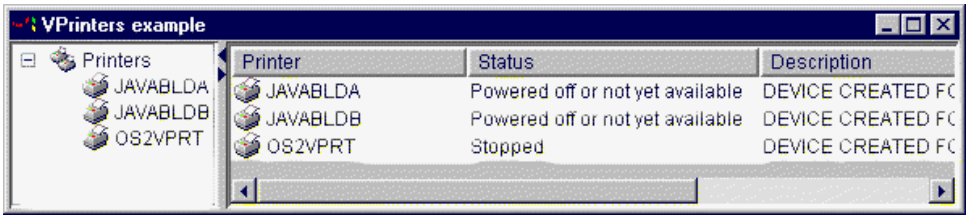
- 使用具有 VIFSDirectory 物件的 AS400TreePane，呈現整合檔案系統目錄階層。圖 3 顯示完成的產品：

圖 3：搭配使用 AS400TreePane 和 VIFSDirectory 物件



- 搭配使用 AS400ExplorerPane 和 VPrinters 物件來呈現列印資源。圖 4 顯示完成的產品：

圖 4：搭配使用 AS400ExplorerPane 和 VPrinters 物件



指令呼叫

指令呼叫 vaccess (GUI) 元件可讓 Java 程式呈現一個按鈕或功能表項目，以呼叫非互動式的伺服器指令。

CommandCallButton 物件代表在按下時會呼叫伺服器指令的按鈕。CommandCallButton 會延伸「Java 基礎類別 (JFC)」JButton 類別，以讓所有按鈕均有一致的外觀及行爲。

同樣地，CommandCallMenuItem 物件代表在選取時，會呼叫伺服器指令的功能表項目。CommandCallMenuItem 類別會擴充 JFC JMenuItem 類別，使所有功能表項目均有一致的外觀與行爲。

若要使用指令呼叫圖形使用者介面，請同時設定系統與指令特性。這些特性可使用建構子或透過 setSystem() 及 setCommand() 方法來設定。

下列範例會建立 CommandCallButton。在執行期間，當按下該按鈕時，它會建立一個叫「FRED」的檔案庫：

```
                        // Create the CommandCallButton
                        // object. Assume that "system" is
                        // an AS400 object created and
                        // initialized elsewhere.  The button
                        // text says "Press Me", and there is
                        // no icon.
  CommandCallButton button = new CommandCallButton ("Press Me", null, system);

                        // Set the command that the button will run.
  button.setCommand ("CRTLIB FRED")；

                        // Add the button to a frame. Assume
                        // that "frame" is a JFrame created
                        // elsewhere.
  frame.getContentPane ().add (button);
```

當執行伺服器指令時，它可能會傳回零或多個伺服器訊息。若要偵測伺服器指令的執行，請使用 addActionCompletedListener() 方法，將 ActionCompletedListener 新增到按鈕或功能表項目中。如此，每當指令執行時，它會傳送一個 ActionCompletedEvent 給所有這類接收程式。接收程式可以使用 getMessageList() 方法，來擷取該指令所產生的任何伺服器訊息。

此範例會新增一個 ActionCompletedListener，來處理該指令所產生的所有伺服器訊息：

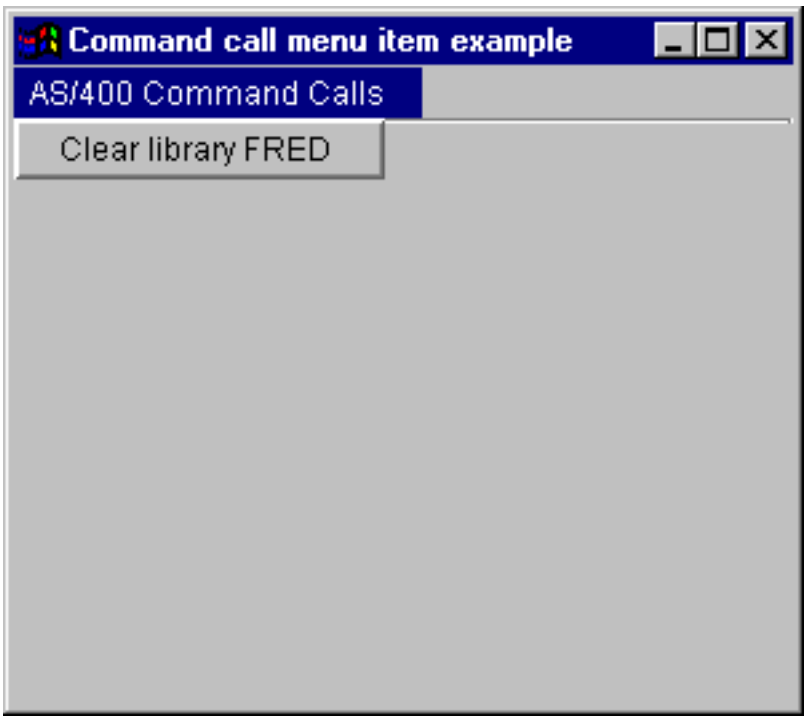
```
                        // Add an ActionCompletedListener that
                        // is implemented using an anonymous
                        // inner class. This is a convenient
                        // way to specify simple event
                        // listeners.

button.addActionCompletedListener (new ActionCompletedListener ()
     {
          public void actionCompleted (ActionCompletedEvent event)
          {
                        // Cast the source of the event to a
                        // CommandCallButton.
               CommandCallButton sourceButton = (CommandCallButton) event.getSource ();

                        // Get the list of server messages
                        // that the command generated.
               AS400Message[] messageList = sourceButton.getMessageList ();

                        // ... Process the message list.
          }
     });
```

範例

此範例顯示如何在應用程式中使用 CommandCallMenuItem。

圖 1 顯示 CommandCall 圖形式使用者介面元件：

圖 1：CommandCall GUI 元件



資料佇列

資料佇列圖形式元件可讓 Java 程式使用任何「Java 基礎類別 (JFC)」圖形文字元件，來讀取或寫入伺服器資料佇列。

DataQueueDocument 和 KeyedDataQueueDocument 類別爲「JFC 文件」介面的實作。這些類別可直接與任何 JFC 圖形文字元件搭配使用。有數種文字元件 (如單一行欄位 (JTextField) 與多行文字區 (JTextArea)) 均可在 JFC 中使用。

資料佇列文件會將文字元件的內容與伺服器資料佇列連結。(文字元件即是用來顯示使用者可選用性編輯的文字的圖形元件。) Java 程式可隨時在文字元件及資料佇列之間讀取及寫入。對**循序**資料佇列使用 DataQueueDocument，對**索引**資料佇列則使用 KeyedDataQueueDocument。

若要使用 DataQueueDocument，請同時設定系統與路徑特性。這些特性可使用建構子或透過 setSystem() 與 setPath() 方法來設定。接著，DataQueueDocument 物件會被「拉進」文字元件，通常是使用文字元件建構子或 setDocument() 方法。 KeyedDataQueueDocuments 也是同樣的運作方式。

下面範例會建立一個 DataQueueDocument，其內容會與資料佇列相關聯：

```
                    // Create the DataQueueDocument
                    // object. Assume that "system" is
                    // an AS400 object created and
                    // initialized elsewhere.
DataQueueDocument dqDocument = new DataQueueDocument (system, "/QSYS.LIB/MYLIB.LIB/MYQUEUE.DTAQ");

                    // Create a text area to present the
                    // document.
JTextArea textArea = new JTextArea (dqDocument);

                    // Add the text area to a frame.
                    // Assume that "frame" is a JFrame
                    // created elsewhere.
frame.getContentPane ().add (textArea);
```

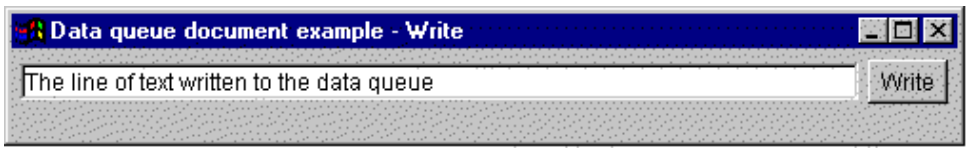
最初，文字元件的內容是空的。使用 read() 或 peek()，以佇列中的下一個登錄填入內容。 使用 write() 將文字元件的內容寫入到資料佇列中。請注意，這些文件只適用於「字串」資料佇列項目。

範例

在應用程式中使用 DataQueueDocument 的範例。

圖 1 顯示在 JTextField 中使用 DataQueueDocument 圖形式使用者介面元件。其中新增了一個按鈕，提供一個 GUI 介面讓使用者將測試欄位的內容寫入資料佇列中。

圖 1：DataQueueDocument GUI 元件



錯誤事件

在大多數情況中，IBM Toolbox for Java GUI 元件會發出錯誤事件，而不是丟出異常。

錯誤事件是內部元件所丟出的異常情況的外層。

您可以提供一錯誤接收器，來處理特殊圖形使用者介面元件所發出的所有錯誤事件。每當丟出一個異常情況，即會呼叫接收器，它可以提供適當的錯誤報告。依據預設值，當發出錯誤事件時，不會採取任何動作。

IBM Toolbox for Java 提供一個稱爲 ErrorDialogAdapter 的圖形式使用者介面元件，每當發生錯誤事件時，它就會自動對使用者顯示一個對話框。

範例

下列範例告訴您如何處理錯誤及定義一個簡單錯誤接受器。

範例：顯示對話框來處理錯誤事件

下列範例將經由顯示對話框，告訴您如何處理錯誤事件：

```
     // All the setup work to lay out a graphical user interface component
     // is done. Now add an ErrorDialogAdapter as a listener to the component.
     // This will report all error events fired by that component through
     // displaying a dialog.


ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (parentFrame);

component.addErrorListener (errorHandler);
```

範例：定義錯誤接收器

您可以撰寫一個自訂錯誤接收器，以不同方式處理錯誤。使用 ErrorListener 介面來完成此工作。

下列範例將告訴您如何定義一個簡單的錯誤接收器，僅將錯誤列印到 System.out：

```
          class MyErrorHandler
          implements ErrorListener
     {
        // This method is invoked whenever an error event is fired.
        public void errorOccurred(ErrorEvent event)
```

```
    {
        Exception e = event.getException ();
        System.out.println ("Error: " + e.getMessage ());
    }
}
```

範例：使用錯誤接收器來處理錯誤事件

下列範例將告訴您如何使用此自訂的處理程式，來處理圖形使用者介面元件的錯誤事件：

```
MyErrorHandler errorHandler = new MyErrorHandler ();
component.addErrorListener (errorHandler);
```

相關參考

第 229 頁的『Vaccess 類別』
Vaccess 套件及其類別已棄用。建議您改爲結合使用 Access 套件及 Java Swing。

第 46 頁的『異常情況』
發生裝置錯誤、實體限制、程式設計錯誤或使用者輸入錯誤時，IBM Toolbox for Java Access 類別會丟出異常。根據發生的錯誤類型而不是出現錯誤的位置來決定異常情況類別。

整合檔案系統

整合檔案系統圖形式使用者介面元件可讓 Java 程式以 GUI 方式，呈現位於伺服器上的整合檔案系統中的目錄及檔案。

可用的元件如下：

- IFSFileSystemView 提供指向 iSeries 整合檔案系統的閘道。
- IFSFileDialog 會呈現一個對話框，以容許使用者經由瀏覽目錄架構來選擇一個目錄並選取一個檔案
- VIFSDirectory 是一種資源，它代表 AS400Panes 所使用的整合檔案系統的目錄。
- IFSTextFileDocument 代表一個要在任何「Java 基礎類別 (JFC)」圖形文字元件中使用的文字檔。
- 若要使用整合檔案系統圖形使用者介面元件，請同時設定系統與路徑或目錄特性。這些特性可使用建構子或透過 setDirectory() (用於 IFSFileDialog) 或 setSystem() 與 setPath() methods (用於 VIFSDirectory 及 IFSTextFileDocument) 來設定。

請設定不同於 "/QSYS.LIB" 的路徑，因爲這個目錄通常很大，在下載內容時會花很長的時間。

IFSFileSystemView:

此類別已棄用，且由 com.ibm.as400.access.IFSSystemView 類別置換。

IFSFileSystemView 提供指向 iSeries 整合檔案系統的閘道，以備建構 javax.swing.JFileChooser 物件時使用。

JFileChooser 是建置用於導覽及選擇檔案之對話框的標準 Java 方式，同時也是 IFSFileDialog 的建議替代方式。

範例：使用 IFSFileSystemView

下列範例會示範 IFSFileSystemView 的使用方式。

```
import com.ibm.as400.access.AS400;
import com.ibm.as400.access.IFSJavaFile;
import com.ibm.as400.vaccess.IFSFileSystemView;
import javax.swing.JFileChooser;
import java.awt.Frame;

// Work with directory /Dir on the system myAS400.
AS400 system = new AS400("myAS400");
```

```
IFSJavaFile dir = new IFSJavaFile(system, "/Dir");
JFileChooser chooser = new JFileChooser(dir, new IFSFileSystemView(system));
Frame parent = new Frame();
int returnVal = chooser.showOpenDialog(parent);
if (returnVal == JFileChooser.APPROVE_OPTION) {
   IFSJavaFile chosenFile = (IFSJavaFile)(chooser.getSelectedFile());
   System.out.println("You selected the file named " +
                      chosenFile.getName());
}
```

檔案對話框:

IFSFileDialog 類別是一個對話框，可讓使用者遍訪位於伺服器上整合檔案系統中的目錄並選取檔案。呼叫程式可設定對話框上的按鈕中的文字。此外，呼叫程式可以使用 FileFilter 物件，讓使用者限制只選擇某些檔案。

如果使用者在對話框中選取檔案，請使用 getFileName() 方法，來取得所選取檔案的名稱。使用 getAbsolutePath() 方法可以取得所選取檔案的完整路徑名稱。

下面範例設置一個具有兩個檔案過濾條件的整合檔案系統檔案對話框：

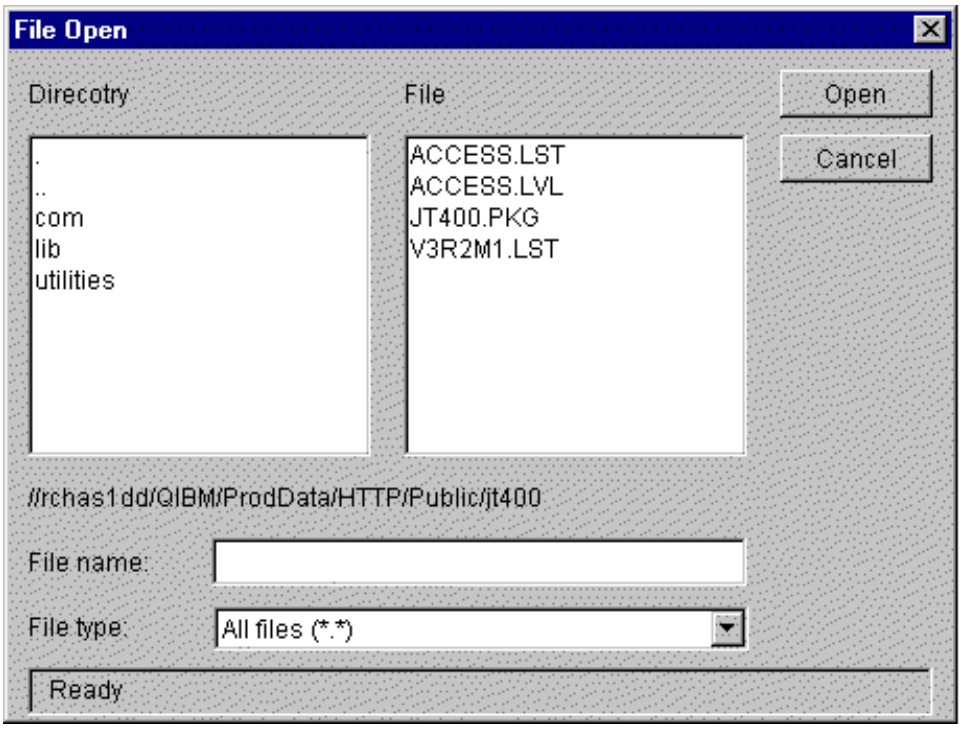
```
                    // Create a IFSFileDialog object
                    // setting the text of the title bar.
                    // Assume that "system" is an AS400
                    // object and "frame" is a JFrame
                    // created and initialized elsewhere.
    IFSFileDialog dialog = new IFSFileDialog (frame, "Select a file", system);

                    // Set a list of filters for the dialog.
                    // The first filter will be used
                    // when the dialog is first displayed.
    FileFilter[] filterList = {new FileFilter ("All files (*.*)", "*.*"),
                               new FileFilter ("HTML files (*.HTML", "*.HTM")};
            // Then, set the filters in the dialog.
    dialog.setFileFilter (filterList, 0);

                    // Set the text on the buttons.
    dialog.setOkButtonText ("Open");
    dialog.setCancelButtonText ("Cancel");

                    // Show the dialog. If the user
// selected a file by pressing the
                    // "Open" button, then print the path
                    // name of the selected file.
     if (dialog.showDialog() == IFSFileDialog.OK)
         System.out.println(dialog.getAbsolutePath());
```

範例

呈現 IFSFileDialog 並列印選擇， 若有的話

圖 1 顯示 IFSFileDialog 圖形式使用者介面元件：

圖 1：IFSFileDialog GUI 元件



AS400Panes 的目錄:

AS400Panes 是一些 GUI 元件，它們會呈現及容許操作一個或多個伺服器資源。VIFSDirectory 物件是一種資源，它代表 AS400Panes 中使用的整合檔案系統的目錄。您可以一起使用 AS400Pane 與 VIFSDirectory 物件，來呈現整合檔案系統的多種檢視方式，並容許使用者導覽、操作及選取目錄與檔案。

若要使用 VIFSDirectory，請同時設定系統與路徑特性。這些特性可使用建構子或透過 setSystem() 與 setPath() 方法來設定。 然後，您可以使用建檔元件或 AS400Pane 的 setRoot() 方法， 將 VIFSDirectory 物件插入 AS400Pane，作爲根目錄。

VIFSDirectory 含有其它一些有用的內容，有助於定義 AS400Panes 中所呈現的目錄及檔案集。使用 setInclude() 指定要出現目錄、檔案或兩者。 使用 setPattern()，經由指定檔案名稱必須符合的型樣，對要顯示的項目設定一個過濾條件。您可以使用萬用字元，例如：「*」及「?」在此型樣中。同樣地，使用 setFilter() 可以設定具有 IFSFileFilter 物件的過濾字元。

當建立 AS400Pane 物件與 VIFSDirectory 物件時，它們會起始設定爲預設狀態。構成根目錄內容的次目錄及檔案尚未載入。若要載入內容，呼叫程式必須在其中一個物件中明確地呼叫 load()方法，起始與伺服器的通訊，以便收集目錄的內容。

在執行期間，使用者可以在任何目錄或檔案上按一下滑鼠右鍵，以便顯示環境功能表來針對那些目錄或檔案執行動作。目錄環境功能表可能包括下列項目：

- **建立檔案** - 在目錄中建立檔案。這將給與檔案預設名稱
- **建立目錄** - 以預設名稱建立子目錄
- **更名** - 變更目錄名稱
- **刪除** - 刪除目錄
- **內容** - 顯示如位置、檔案和子目錄數，及修正日期等內容

檔案環境功能表可能包括下列項目：

- **編輯** - 在不同視窗中編輯文字檔
- **檢視** - 在不同視窗中檢視文字檔
- **更名** - 變更檔案名稱
- **刪除** - 刪除檔案
- **內容** - 顯示如位置、大小、修改日期及屬性等內容

使用者僅能讀取或寫入它們有權使用的目錄及檔案。此外，呼叫程式能在窗格中使用 setAllowActions() 方法，阻止使用者執行動作。

下面範例會建立一個 VIFSDirectory，並在 AS400ExplorerPane 中呈現它：

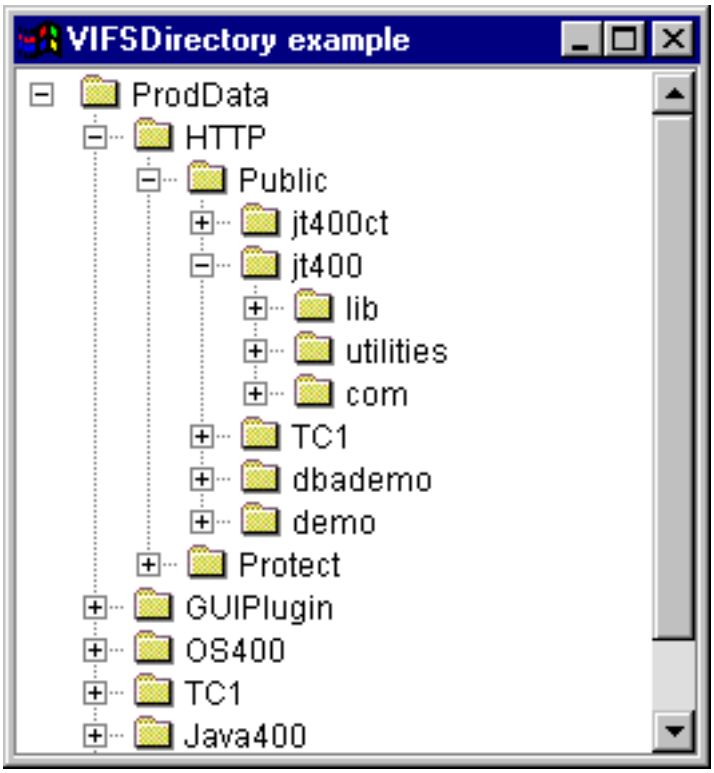
```
                    // Create the VIFSDirectory object.
                    // Assume that "system" in an AS400
                    // object created and initialized
                    // elsewhere.
    VIFSDirectory root = new VIFSDirectory (system, "/DirectoryA/DirectoryB");

                    // Create and load an AS400ExplorerPane object.
    AS400ExplorerPane explorerPane = new AS400ExplorerPane (root);

explorerPane.load ();

                    // Add the explorer pane to a frame.
                    // Assume that "frame" is a JFrame
                    // created elsewhere.
    frame.getContentPane ().add (explorerPane);
```

範例

使用具有 VIFSDirectory 物件的 AS400TreePane 來呈現整合檔案系統目錄架構

圖 1 顯示 VIFSDirectory 圖形式使用者介面元件：

圖 1：VIFSDirectory GUI 元件



IFSTextFileDocument:

文字檔文件可讓 Java 程式使用任何「Java 基礎類別 (JFC)」圖形文字元件，來編輯或檢視伺服器上的整合檔案系統中的文字檔。(文字元件即是用來顯示使用者可選用性編輯的文字的圖形元件。)

IFSTextFileDocument 類別是「JFC 文件」介面的實作。它可以直接與任何 JFC 圖形文字元件搭配使用。 有數種文字元件 (如單一行欄位 (JTextField) 與多行文字區 (JTextArea)) 均可在 JFC 中使用。

文字檔文件會使文字元件的內容與文字檔產生關聯。Java 程式可隨時在文字元件與文字檔之間進行載入及儲存。

若要使用 IFSTextFileDocument，請同時設定系統與路徑特性。這些特性可使用建構子或透過 setSystem() 與 setPath() 方法來設定。然後， IFSTextFileDocument 物件會被「拉進」文字元件，通常是使用文字元件建構子或 setDocument() 方法。

最初，文字元件的內容是空的。使用 load() 從文字檔中載入內容。使用 save() 將文字元件的內容儲存到文字檔中。

下面範例會建立及載入 IFSTextFileDocument：

```
                        // Create and load the
                        // IFSTextFileDocument object. Assume
                        // that "system" is an AS400 object
                        // created and initialized elsewhere.
    IFSTextFileDocument ifsDocument = new IFSTextFileDocument (system, "/DirectoryA/MyFile.txt");

ifsDocument.load ();

                        // Create a text area to present the
                        // document.
    JTextArea textArea = new JTextArea (ifsDocument);
```

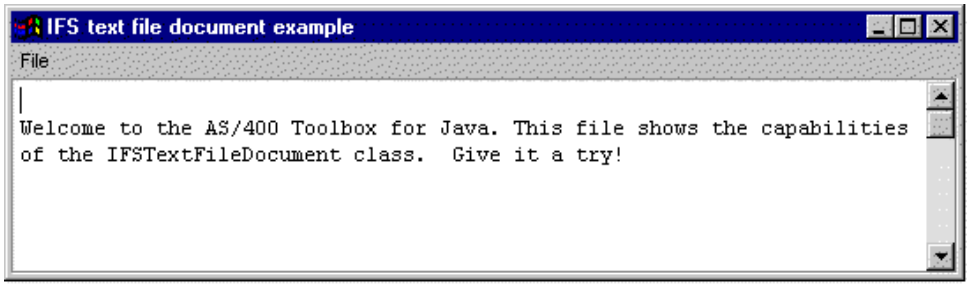
```
                    // Add the text area to a frame.
                    // Assume that "frame" is a JFrame
                    // created elsewhere.
    frame.getContentPane ().add (textArea);
```

範例

在 JTextPane 中呈現 IFSTextFileDocument。

圖 1 顯示 IFSTextFileDocument 圖形式使用者介面元件：

圖 1：IFSTextFileDocument 範例



VJavaApplicationCall 類別

VJavaApplicationCall 類別可讓您使用圖形式使用者介面 (GUI)，自用戶端執行伺服器上的 Java 應用程式。

GUI 是具有兩個區段的畫面。上面的區段是輸出視窗，顯示 Java 程式寫入標準輸出及標準錯誤的輸出。下面的區段是輸入欄位，使用者可在其中輸入 Java 環境、要使用參數執行的 Java 程式，以及 Java 程式透過標準輸入接收的輸入。如需相關資訊，請參閱 Java 指令選項。

例如，此程式碼可以爲 Java 程式建立下列 GUI。

VJavaApplicationCall 是一個您可以從 Java 程式呼叫的類別。然而，IBM Toolbox for Java 也提供一個公用程式，該公用程式是完整的 Java 應用程式，可以用來從工作站呼叫 Java 程式。請參閱 RunJavaApplication 類別，以取得其餘相關資訊。

JDBC 類別

JDBC 圖形式使用者介面元件可讓 Java 程式呈現不同的檢視及控制方式，以便使用 SQL (結構化查詢語言) 陳述式及查詢來存取資料庫。

可用的元件如下：

- SQLStatementButton 和 SQLStatementMenuItem 是按鈕或功能表項目，當您按一下或選取它時，會發出一個 SQL 陳述式。
- SQLStatementDocument 是一份可與任何「Java 基礎類別 (JFC)」圖形文字元件搭配使用以發出 SQL 陳述式的文件。
- SQLResultSetFormPane 會以表單方式呈現 SQL 查詢的結果
- SQLResultSetTablePane 會以表格呈現 SQL 查詢的結果
- SQLResultSetTableModel 會管理表格中 SQL 查詢的結果
- SQLQueryBuilderPane 會呈現可動態建置 SQL 查詢的互動式工具

所有 JDBC 圖形使用者介面元件均會使用 JDBC 驅動程式與資料庫進行通訊。必須已用 JDBC驅動程式管理程式登記了 JDBC 驅動程式，以便使這些元件的任一個均可運作。下列範例會登記 IBM Toolbox for Java JDBC 驅動程式：

```
                        // Register the JDBC driver.

DriverManager.registerDriver (new com.ibm.as400.access.AS400JDBCDriver
());
```

SQL 連接

SQLConnection 物件代表使用 JDBC 的資料庫連線。**SQLConnection 物件會與所有 JDBC圖形使用者介面元件搭配使用**。

若要使用 SQLConnection，請使用建構子或 setURL() 來設定 URL 內容。這會識別對其產生連接的資料庫。可以設定其它選用性特性：

- 使用 setProperties() 可以指定一組 JDBC 連線內容。
- 使用 setUserName() 可以指定連線的使用者名稱。
- 使用 setPassword() 可以指定連線的密碼。

當建立 SQLConnection 物件時，並不會產生實際的資料庫連線。相反地，它是當您呼叫 getConnection() 時建立連線。正常情況下，JDBC 圖形使用者介面元件會自動呼叫此方法，但也可以隨時呼叫，以便控制產生連接的時間。

下面範例會建立及起始設定 SQLConnection 物件：

```
                         // Create an SQLConnection object.

SQLConnection connection = new SQLConnection ();

                        // Set the URL and user name properties of the connection.
     connection.setURL ("jdbc:as400://MySystem");
     connection.setUserName ("Lisa");
```

SQLConnection 物件可對多個 JDBC 圖形使用者介面元件使用。所有如此的元件將使用同一個連接，因而可增進效能及資源使用。另一種方法，每一 JDBC 圖形使用者介面元件均可使用不同的 SQL物件。有時候使用個別連接是必需的，以便在不同異動中發出 SQL 陳述式。

當不再需要連線時，請使用 close() 來關閉 SQLConnection 物件。這會同時釋放用戶端與伺服器中的 JDBC 資源。

按鈕與功能表項目:

SQLStatementButton 代表一個按鈕，當您按下它時，它會發出一個 SQL (Structured Query Language) 陳述式。SQLStatementButton 類別會延伸「Java 基礎類別 (JFC)」JButton 類別，以讓所有按鈕均有一致的外觀及行爲。

同樣地，SQLStatementMenuItem 物件代表在選取時會發出 SQL 陳述式的功能表項目。SQLStatementMenuItem 類別會擴充 JFC JMenuItem 類別，以便所有功能表項目均有一致的外觀與行爲。

若要使用這兩個類別之一，請同時設定連接與 SQLStatement 特性。這些特性可使用建構子或透過 setConnection() 與 setSQLStatement() 方法來設定。

下面範例會建立 SQLStatementButton。在執行期間，若按下此按鈕，它會刪除表格中的所有記錄：

```
                        // Create an SQLStatementButton object.
                        // The button text says "Delete All",
                        // and there is no icon.
```

```
SQLStatementButton button = new SQLStatementButton ("Delete All");

                    // Set the connection and SQLStatement
                    // properties.  Assume that "connection"
                    // is an SQLConnection object that is
                    // created and initialized elsewhere.
button.setConnection (connection);
button.setSQLStatement ("DELETE FROM MYTABLE");

                    // Add the button to a frame. Assume
                    // that "frame" is a JFrame created
                    // elsewhere.
frame.getContentPane ().add (button);
```

在發出 SQL 陳述式後，請使用 getResultSet(), getMoreResults(), getUpdateCount() 或 getWarnings() 來擷取結果。

SQLStatementDocument 類別:

SQLStatementDocument 類別是「Java 基礎類別 (JFC) 文件」介面的實作。它可以直接與任何 JFC 圖形文字元件搭配使用。 有數種文字元件 (如單一行欄位 (JTextField) 與多行文字區 (JTextArea)) 均可在 JFC 中使用。SQLStatementDocument 物件會使文字元件的內容與與 SQLConnection 物件產生關聯。Java 程式可以隨時執行文件內容中所含有的 SQL 陳述式，然後處理結果 (若有的話)。

若要使用 SQLStatementDocument，您必須設定 connection 特性。經由使用建構子或 setConnection() 方法來設定此特性。然後，SQLStatementDocument 物件會被「拉進」文字元件，通常是使用文字元件建構子或 setDocument() 方法。您可以隨時使用 execute() 來執行文件所包含的 SQL 陳述式。

下面範例會在 JTextField 中建立 SQLStatementDocument：

```
                    // Create an SQLStatementDocument
                    // object. Assume that "connection"
                    // is an SQLConnection object that is
                    // created and initialized elsewhere.
                    // The text of the document is
                    // initialized to a generic query.
SQLStatementDocument document = new SQLStatementDocument (connection, "SELECT * FROM QIWS.QCUSTCDT");

                    // Create a text field to present the
                    // document.
JTextField textField = new JTextField ();

textField.setDocument (document);

                    // Add the text field to a frame.
                    // Assume that "frame" is a JFrame
                    // created elsewhere.
frame.getContentPane ().add (textField);

                    // Run the SQL statement that is in
                    // the text field.
document.execute ();
```

發出 SQL 陳述式之後，請使用 getResultSet()、getMoreResults()、getUpdateCount() 或 getWarnings() 來擷取結果。

SQLResultSetFormPane 類別:

SQLResultSetFormPane 會將 SQL (結構化查詢語言) 查詢的結果呈現在套表中。表單會一次顯示一筆記錄，並提供一些按鈕，容許使用者向前捲動、向後捲動到第一筆或最後一筆記錄，或是復新結果的檢視畫面。

若要使用 SQLResultSetFormPane，請設定連接與查詢特性。這些內容可以使用建構子或 setConnection() 和 setQuery() 方法來設定。使用 load() 以執行查詢並呈現結果集中的第一筆記錄。當您不再需要結果時，請呼叫 close() 來確定關閉結果集。

下面範例會建立 SQLResultSetFormPane 物件，並將它新增到頁框中：

```
                    // Create an SQLResultSetFormPane
                    // object. Assume that "connection"
                    // is an SQLConnection object that is
                    // created and initialized elsewhere.
SQLResultSetFormPane formPane = new SQLResultSetFormPane (connection, "
                                  SELECT * FROM QIWS.QCUSTCDT");
                    // Load the results.
formPane.load ();

                    // Add the form pane to a frame.
                    // Assume that "frame" is a JFrame
                    // created elsewhere.
frame.getContentPane ().add (formPane);
```

SQLResultSetTablePane 類別:

SQLResultSetTablePane 會在表格中呈現 SQL (結構化查詢語言) 查詢的結果。表格中的每一列顯示來自結果集的記錄，而每一欄則顯示欄位。

若要使用 SQLResultSetTablePane，請設定連接與查詢特性。這些內容可以使用建構子或 setConnection() 和 setQuery() 方法來設定。使用 load() 以執行查詢並將結果呈現在表格中。當您不再需要結果時，請呼叫 close() 來確定關閉結果集。

下面範例會建立 SQLResultSetTablePane 物件，並將它新增到頁框中：

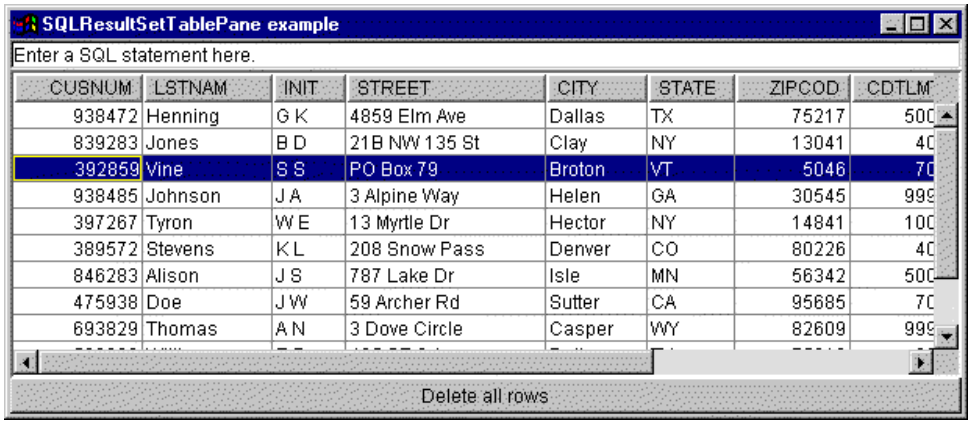
```
                    // Create an SQLResultSetTablePane
                    // object. Assume that "connection"
                    // is an SQLConnection object that is
                    // created and initialized elsewhere.
SQLResultSetTablePane tablePane = new SQLResultSetTablePane (connection,
                                    "SELECT * FROM QIWS.QCUSTCDT");
                    // Load the results.
tablePane.load ();

                    // Add the table pane to a frame.
                    // Assume that "frame" is a JFrame
                    // created elsewhere.
frame.getContentPane ().add (tablePane);
```

範例

呈現會顯示表格內容的 SQLResultSetTablePane。此範例使用 SQLStatementDocument (用文字表示的下列影像，「在此輸入 SQL 陳述式」)，它容許使用者鍵入任何的 SQL 陳述式，及 SQLStatementButton (用文字表示，「刪除全部的列」)，它容許使用者從表格中刪除全部的列。

圖 1 顯示 SQLResultSetTablePane 圖形式使用者介面元件：

圖 1：SQLResultSetTablePane GUI 元件



SQLResultSetTableModel 類別:

SQLResultSetTablePane 是使用模型-概略表-控制器範例來實施，在此資料與使用者介面被分成不同的類別。此實作方式會將 SQLResultSetTableModel 與「Java 基礎類別 (JFC)」的 JTable 整合。SQLResultSetTableModel 類別會管理查詢的結果， 而 JTable 則以圖形方式顯示結果並處理使用者互動。

SQLResultSetTablePane 會提供足夠的功能來符合大部分的需求。不過，如果呼叫程式需要更進一步控制 JFC 元件，則呼叫程式可以直接使用 SQLResultSetTableModel，並提供與不同圖形使用者介面元件的自訂整合。

若要使用 ResAltSotAoTebSAMmdAS，請設定連接與查詢特性。這些內容可以使用建構子或 setConnection() 和 setQuery() 方法來設定。使用 load() 以執行查詢並載入結果。當您不再需要結果時，請呼叫 close() 來確定關閉結果集。

下面範例會建立 SQLResultSetTableModel 物件，並以 JTable 呈現它：

```
                    // Create an SQLResultSetTableModel
                    // object. Assume that "connection"
                    // is an SQLConnection object that is
                    // created and initialized elsewhere.
SQLResultSetTableModel tableModel = new SQLResultSetTableModel (connection,
                                    "SELECT * FROM QIWS.QCUSTCDT");
                    // Load the results.
tableModel.load ();

                    // Create a JTable for the model.
JTable table = new JTable (tableModel);

                    // Add the table to a frame.  Assume
                    // that "frame" is a JFrame created
                    // elsewhere.
frame.getContentPane ().add (table);
```

SQL 查詢建置器:

SQLQueryBuilderPane 會呈現一個互動式工具，可用來動態建置 SQL 查詢。

若要使用 SQLQueryPane，請設定連接特性。您可以使用建構子或 setConnection() 方法來設定這個內容。使用 load() 可以載入查詢建置器圖形式使用者介面所需的資料。使用 getQuery() 可以取得使用者已建立的 SQL 查詢。

下面範例會建立 SQLQueryBuilderPane 物件，並將它新增到頁框中：

```
                    // Create an SQLQueryBuilderPane
                    // object. Assume that "connection"
                    // is an SQLConnection object that is
```

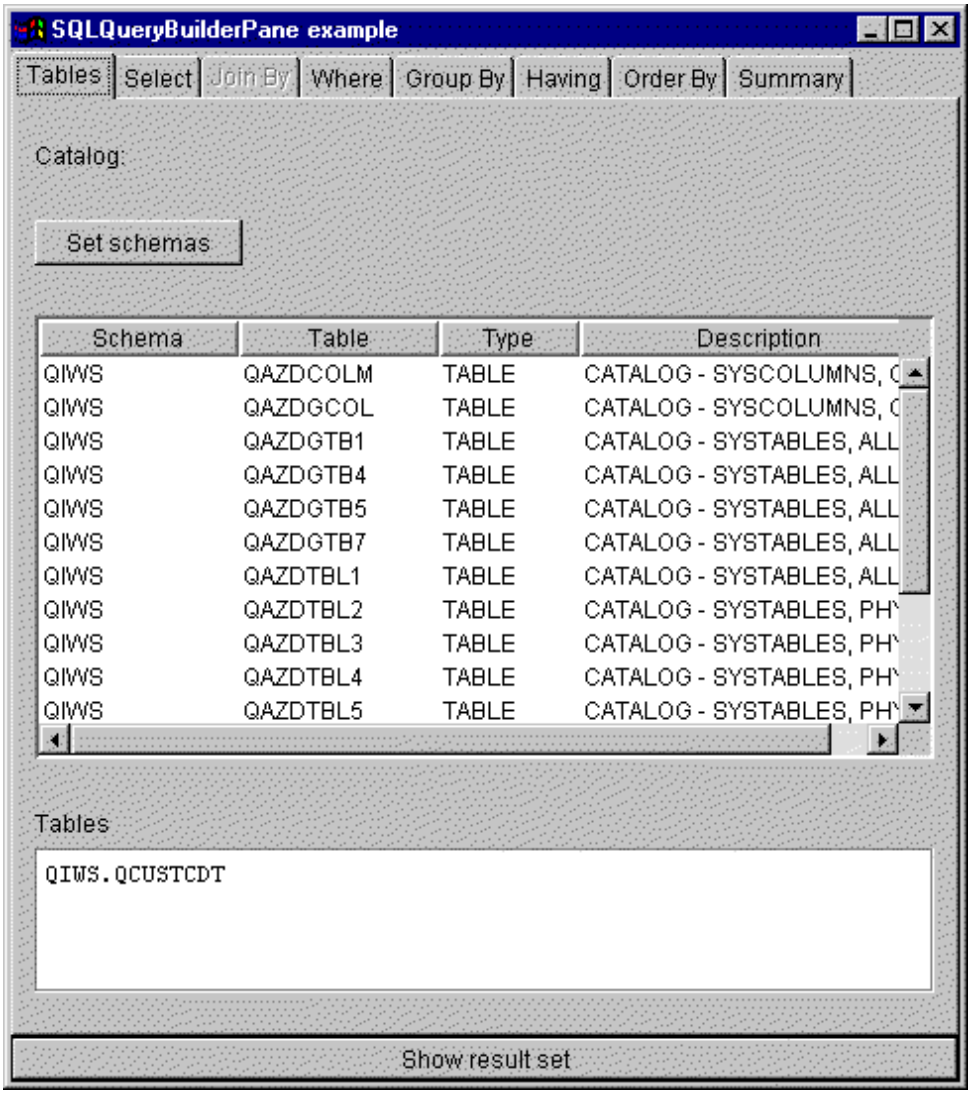
```
                    // created and initialized elsewhere.
SQLQueryBuilderPane queryBuilder = new SQLQueryBuilderPane (connection);

                    // Load the data needed for the query
                    // builder.
queryBuilder.load ();

                    // Add the query builder pane to a
                    // frame. Assume that "frame" is a
                    // JFrame created elsewhere.
frame.getContentPane ().add (queryBuilder);
```

範例

呈現 SQLQueryBuilderPane 及按鈕當按一下按鈕時，會在另一個頁框中呈現 SQLResultSetFormPane 中查詢的結果。

圖 1 顯示 SQLQueryBuilderPane 圖形式使用者介面元件：

圖 1：SQLQueryBuilderPane GUI 元件



工作

工作 vaccess (GUI) 元件可讓 Java 程式在 GUI 中呈現伺服器工作清單及工作日誌訊息。

可用的元件如下：

- VJobList 物件是一種資源，它會呈現 AS400Panes 中所使用的伺服器工作清單。
- VJob 物件是一種資源，它會呈現要在 AS400Panes 中使用的工作日誌的訊息清單。

您可以合併使用 AS400Panes、VJobList 物件與 VJob 物件，以多種檢視方式呈現工作清單或工作日誌。

若要使用 VJobList，請設定系統、名稱、號碼及使用者特性。這些內容可以使用建構子或透過 setSystem()、setName()、setNumber() 和 setUser() 內容來設定。

若要使用 VJob，請設定系統特性。此內容可以使用建構子或透過 setSystem() 方法來設定。

VJobList 或 VJob 物件接著將「插入」AS400Pane 來作爲根，方式是使用窗格的建構子或 setRoot() 方法。

VJobList 另有一些特性有助於定義 AS400Panes 中所呈現的工作集。使用 setName()，可以指定僅顯示具有特定名稱的工作。使用 setNumber()，可以指定僅顯示具有特定編號的工作。同樣地，使用 setUser()，可以指定僅顯示特定使用者的工作。

當建立 AS400Pane、VJobList 與 VJob 物件時，它們會起始設定爲預設狀態。建立時，不會載入工作或工作日誌訊息的清單。若要載入內容，呼叫程式必須在這兩個物件之一中以明確方式呼叫 load()方法。這會起始與伺服器系統的通訊，來收集清單的內容。

在執行時間，以滑鼠右鍵按一下工作、工作清單或工作日誌訊息，以顯示捷徑功能表。從捷徑功能表中選取**內容**，針對選取的物件執行動作：

- 工作 - 使用內容，例如類型和狀態。您也可以變更部分內容的值。
- 工作清單 - 使用內容，例如名稱、號碼及使用者內容。您也可以變更清單的內容。
- 工作日誌訊息 - 顯示內容，例如：全文、嚴重性及傳送時間。

使用者僅能存取他們有權使用的工作。此外，Java 程式可在窗格中使用 setAllowActions() 方法，以防止使用者執行動作。

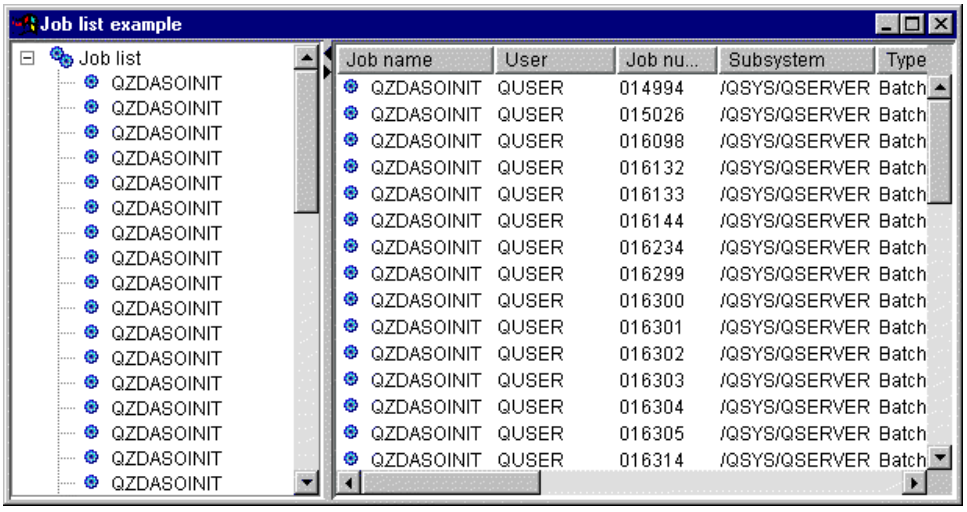
下面範例會建立 VJobList，並在 AS400ExplorerPane 中予以呈現：

```
                        // Create the VJobList object. Assume
                        // that "system" is an AS400 object
                        // created and initialized elsewhere.
    VJobList root = new VJobList (system);

                        // Create and load an
                        // AS400ExplorerPane object.
    AS400ExplorerPane explorerPane = new AS400ExplorerPane (root);

explorerPane.load ();

                        // Add the explorer pane to a frame.
                        // Assume that "frame" is a JFrame
                        // created elsewhere.
    frame.getContentPane ().add (explorerPane);
```

範例

此 VJobList 範例會呈現已填入工作清單的 AS400ExplorerPane。此清單會顯示系統中具有相同工作名稱的工作。

下面影像顯示 VJobList 圖形使用者介面元件：



Vaccess message 類別

訊息圖形式使用者介面元件可讓 Java 程式在 GUI 中呈現伺服器訊息清單。

可用的元件如下：

- 訊息清單物件是一種資源， 代表在 AS400Panes 中所使用的訊息清單。這是針對指令或程式呼叫所產生的訊息清單。
- 訊息佇列物件是一種資源，代表 AS400Ppanes 中所使用的伺服器訊息佇列中的訊息。

AS400Panes 是圖形式使用者介面元件，它們會呈現及容許操作一個或多個伺服器資源。VMessageList 和 VMessageQueue 物件是代表 AS400Panes 中的伺服器訊息清單的資源。

您可以合併使用 AS400Pane、VMessageList 與 VMessageQueue 物件，以不同檢視方式呈現訊息清單，並允許使用者選取訊息及對它們執行動作。

VMessageList 類別:

VMessageList 物件是一種資源，代表 AS400Panes 中所使用的訊息清單。這是針對指令或程式呼叫所產生的訊息清單。下列方法會傳回訊息清單：

- CommandCall.getMessageList()
- CommandCallButton.getMessageList()
- CommandCallMenuItem.getMessageList()
- ProgramCall.getMessageList()
- ProgramCallButton.getMessageList()
- ProgramCallMenuItem.getMessageList()

若要使用 VMessageList，請設定 messageList 特性。這個內容可以使用建構子或透過 setMessageList() 方法來設定。接著使用 AS400Pane 的建構子或 setRoot() 方法，將 VMessageList 物件「插入」 AS400Pane 中作爲根。

當建立 AS400Pane 與 VMessageList 物件時，它們會起始設定爲預設狀態。建立時不會載入訊息清單。若要載入內容，呼叫程式必須在這兩個物件之一中以明確方式呼叫 load() 方法。

在執行期間，使用者可以滑鼠右鍵按一下訊息來顯示環境功能表，以對它執行動作。訊息環境功能表可能包括名爲**內容**的項目，它會顯示諸如嚴重性、類型及日期等內容。

呼叫程式可以經由在窗格中使用 setAllowActions() 方法，阻止使用者執行動作。

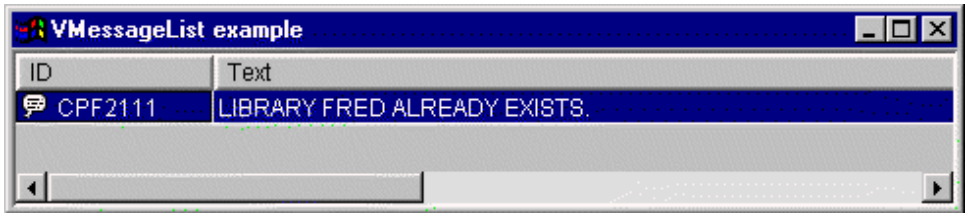
下面範例會建立 VMessageList 供指令呼叫所產生的訊息使用， 並在 AS400DetailsPane 中呈現：

```
                        // Create the VMessageList object.
                        // Assume that "command" is a
                        // CommandCall object created and run
                        // elsewhere.

VMessageList root = new VMessageList (command.getMessageList ());

                        // Create and load an AS400DetailsPane
                        // object.
     AS400DetailsPane detailsPane = new AS400DetailsPane (root);

detailsPane.load ();

                        // Add the details pane to a frame.
                        // Assume that "frame" is a JFrame
                        // created elsewhere.
     frame.getContentPane ().add (detailsPane);
```

範例

呈現使用 AS400DetailsPane 的指令呼叫，其中包含 VMessageList 物件所產生的訊息清單。圖 1 顯示 VMessageList 圖形式使用者介面元件：

圖 1：VMessageList GUI 元件



VMessageQueue 類別:

VMessageQueue 物件是一種資源，代表 AS400Panes 中所使用的伺服器訊息佇列中的訊息。

若要使用 VMessageQueue，請設定系統與路徑特性。您可以使用建構子或透過 setSystem() 和 setPath() 方法來設定這些內容。接著使用 AS400Pane 的建構子或 setRoot() 方法， 將 VMessageQueue 物件「插入」AS400Pane 中作爲根。

VMessageQueue 另有一些其它有用的內容，可用來定義 AS400Panes 所呈現的訊息集。 使用 setSeverity() 來指定出現訊息的嚴重性。使用 setSelection() 來指定出現的訊息類型。

當建立 AS400Pane 與 VMessageQueue 物件時，它們會起始設定爲預設狀態。建立時不會載入訊息清單。若要載入內容，呼叫程式必須在這兩個物件其中之一，以明確方式呼叫 load() 方法。這會起始與伺服器系統的通訊，來收集清單的內容。

在執行期間，使用者可以滑鼠右鍵按一下訊息或訊息佇列來顯示環境功能表，以對它執行動作。訊息佇列的環境功能表可能包括下列項目：

- **清除** - 清除訊息佇列

- **內容** - 允許使用者設定嚴重性和選項內容。這可用來變更清單的內容

下列是可對訊息佇列中的訊息使用的動作：

- **移除** - 從訊息佇列中移除訊息
- **回答** - 回覆查詢訊息
- **內容** - 顯示如嚴重性、類型及日期等內容

當然，使用者僅能存取他們有權使用的訊息佇列。此外，呼叫程式能在窗格中使用 setAllowActions() 方法，阻止使用者執行動作。

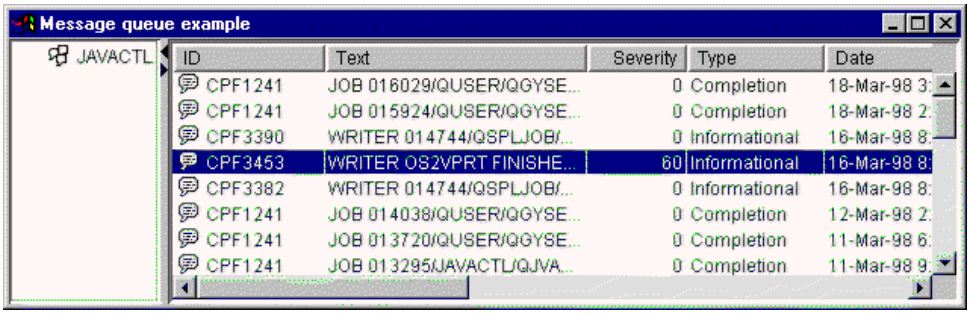
下面範例會建立 VMessageQueue 並在 AS400ExplorerPane 中呈現它：

```
                    // Create the VMessageQueue object.
                    // Assume that "system" is an AS400
                    // object created and initialized
                    // elsewhere.
    VMessageQueue root = new VMessageQueue (system, "/QSYS.LIB/MYLIB.LIB/MYMSGQ.MSGQ");

                    // Create and load an
                    // AS400ExplorerPane object.
    AS400ExplorerPane explorerPane = new AS400ExplorerPane (root);

explorerPane.load ();

                    // Add the explorer pane to a frame.
                    // Assume that "frame" is a JFrame
                    // created elsewhere.
    frame.getContentPane ().add (explorerPane);
```

範例

使用 AS400ExplorerPane，其中包括 VMessageQueue 物件，呈現訊息佇列中的訊息清單。圖 1 顯示 VMessageQueue 圖形式使用者介面元件：

圖 1：VMessageQueue GUI 元件



許可權類別

您可以透過 VIFSFile 和 VIFSDirectory 類別，在圖形式使用者介面 (GUI) 中使用許可權類別。許可權已被新增作爲這些類別的每一個的動作。

下面範例告訴您「許可權」如何搭配 VIFSDirectory 類別一起使用：

```
    // Create AS400 object
    AS400 as400 = new AS400();

    // Create an IFSDirectory using the system name
```

```
// and the full path of a QSYS object
VIFSDirectory directory = new VIFSDirectory(as400,
                                  "/QSYS.LID/testlib1.lib");

    // Create as explorer Pane
    AS400ExplorerPane pane = new AS400ExplorerPane((VNode)directory);

    // Load the information
    pane.load();
```

Vaccess print 類別

Vaccess 套件中的下列元件可讓 Java 程式，在圖形式使用者介面中呈現伺服器列印資源的清單。

- VPrinters 物件是一種資源， 它代表要在 AS400Panes 中使用的印表機清單。
- VPrinter 物件是一種資源， 它代表要在 AS400Panes 中使用的印表機及其排存檔。
- VPrinterOutput 物件是一種資源， 它代表要在 AS400Panes 中使用的排存檔清單。
- SpooledFileViewer 物件爲以視覺化的方式呈現排存檔的資源。

AS400Panes 是一些 GUI 元件，它們會呈現及容許操作一個或多個伺服器資源。VPrinters、VPrinter 及 VPrinterOutput 物件爲資源，代表 AS400Panes 中的伺服器列印資源的清單。

您可以合併使用 AS400Pane、VPrinters、VPrinter 與 VPrinterOutput 物件，以呈現列印資源的不同檢視方式，以及容許使用者選取它們並對它們執行作業。

VPrinters 類別:

VPrinters 物件是一種資源，它代表要在 AS400Panes 中使用的印表機清單。

若要使用 VPrinters 物件，請設定系統特性。此內容可以使用建構子或透過 setSystem() 方法來設定。然後，即會使用窗格的建構子或 setRoot() 方法， 將 VPrinters 物件「插入」AS400Pane 作爲根。

VPrinters 物件有另一種有用的內容，可用來定義 AS400Panes 中所呈現的印表機集。使用 setPrinterFilter() 來指定過濾字元，以定義應該出現的印表機。

當建立 AS400Pane 與 VPrinters 物件時，它們會起始設定爲預設狀態。尚未載入印表機清單。若要載入內容，呼叫程式必須在這兩個物件之一中以明確方式呼叫 load() 方法。

在執行期間，使用者可以在任何印表機清單或印表機上按一下滑鼠右鍵，以便顯示環境功能表來針對它們執行動作。排存檔清單環境功能表可以包括名爲**內容**的項目，讓使用者設定印表機過濾內容，來變更清單的內容。

印表機環境功能表可能包括下列項目：

- **保留** - 保留印表機
- **釋放** - 釋放印表機
- **啓動** - 啓動印表機
- **停止** - 停止印表機
- **可使用** - 使印表機可用
- **不可使用** - 使印表機無法使用
- **內容** - 顯示印表機的內容並容許使用者設定過濾條件

使用者僅能存取他們有權使用的印表機。此外，呼叫程式能在窗格中使用 setAllowActions() 方法，阻止使用者執行動作。

下面範例會建立 VPrinters 物件並在 AS400TreePane 中呈現它

```
                        // Create the VPrinters object.
                        // Assume that "system" is an AS400
                        // object created and initialized
                        // elsewhere.

VPrinters root = new VPrinters (system);

                        // Create and load an AS400TreePane
                        // object.
     AS400TreePane treePane = new AS400TreePane (root);

treePane.load ();

                        // Add the tree pane to a frame.
                        // Assume that "frame" is a JFrame
                        // created elsewhere.
     frame.getContentPane ().add (treePane);
```

範例

搭配使用 AS400ExplorerPane 與 VPrinters 物件，來呈現列印資源。圖 1 顯示 VPrinters 圖形式使用者介面元件：

圖 1：VPrinters GUI 元件

VPrinter 類別:

VPrinter 物件是一種資源，它代表要在 AS400Panes 中使用的伺服器印表機及其排存檔。

若要使用 VPrinter，請設定印表機特性。此內容可以使用建構子或透過 setPrinter() 方法來設定。然後，即會使用窗格的建構子或 setRoot() 方法， 將 VPrinter 物件「插入」AS400Pane 中作爲根。

當建立 AS400Pane 與 VPrinter 物件時，它們會起始設定爲預設狀態。在建立期間，不會載入印表機的屬性與排存檔清單。

若要載入內容，呼叫程式必須在這兩個物件其中之一，以明確方式呼叫 load() 方法。這會起始與伺服器系統的通訊，來收集清單的內容。

在執行期間，使用者可以滑鼠右鍵按一下任何印表機或排存檔，以便顯示環境功能表來針對它們執行動作。訊息佇列的環境功能表可能包括下列項目：

- **保留** - 保留印表機
- **釋放** - 釋放印表機
- **啓動** - 啓動印表機
- **停止** - 停止印表機
- **可使用** - 使印表機可用
- **不可使用** - 使印表機無法使用

- **內容** - 顯示印表機的內容並容許使用者設定過濾條件

針對印表機所列示的排存檔的環境功能表可能包括下列項目：

- **回答** - 回答排存檔
- **保留** - 保留排存檔
- **釋放** - 釋放排存檔
- **列印下一個** - 列印下一個排存檔
- **傳送** - 傳送排存檔
- **移動** - 移動排存檔
- **刪除** - 刪除排存檔
- **內容** - 顯示排存檔的許多內容，並容許使用者變更其中的某些內容

使用者僅能存取他們有權使用的印表機與排存檔。此外，呼叫程式能在窗格中使用 setAllowActions() 方法，阻止使用者執行動作。

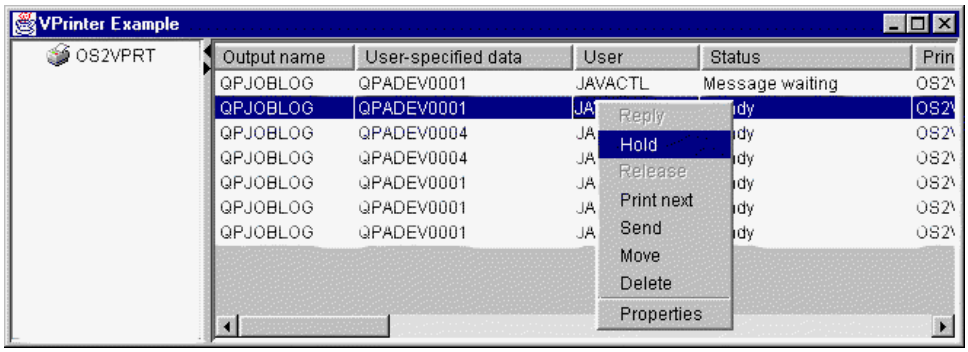
下面範例會建立 VPrinter 並在 AS400ExplorerPane 中呈現它：

```
                    // Create the VPrinter object.
                    // Assume that "system" is an AS400
                    // object created and initialized
                    // elsewhere.
    VPrinter root = new VPrinter (new Printer (system, "MYPRINTER"));

                    // Create and load an
                    // AS400ExplorerPane object.
    AS400ExplorerPane explorerPane = new AS400ExplorerPane (root);

explorerPane.load ();

                    // Add the explorer pane to a frame.
                    // Assume that "frame" is a JFrame
                    // created elsewhere.
    frame.getContentPane ().add (explorerPane);
```

範例

使用具有 VPrinter 物件的 AS400ExplorerPane 來呈現列印資源。圖 1 顯示 VPrinter 圖形式使用者介面元件：

圖 1：VPrinter GUI 元件



VPrinterOutput 類別:

VPrinterOutput 物件是一種資源，它代表可在 AS400Panes 中使用的伺服器上的排存檔清單。

若要使用 VPrinterOutput 物件，請設定系統特性。這些內容可使用建構子或透過 setSystem() 方法來設定。接著使用 AS400Pane 的建構子或 setRoot() 方法， 將 VPrinterOutput 物件「插入」AS400Pane 中作爲根。

VPrinterOutput 物件含有其它有用的內容，有助於定義 AS400Panes 中所呈現的排存檔集。使用 setFormTypeFilter() 來指定應該出現的套表類型。使用 setUserDataFilter() 來指定應該出現的使用者資料。最後，使用 setUserFilter() 來指定應該出現的使用者排存檔。

當建立 AS400Pane 與 VPrinterOutput 物件時，它們會起始設定爲預設狀態。在建立期間，不會載入排存檔清單。若要載入內容，呼叫程式必須在這兩個物件其中之一，以明確方式呼叫 load() 方法。這會起始與伺服器系統的通訊，來收集清單的內容。

在執行期間，使用者可以在任何排存檔或排存檔清單上按一下滑鼠右鍵，以便顯示環境功能表來針對它們執行動作。排存檔清單環境功能表可以包括名爲**內容**的項目，讓使用者設定過濾內容，來變更清單的內容。

排存檔環境功能表可能包括下列項目：

- **回答** - 回答排存檔
- **保留** - 保留排存檔
- **釋放** - 釋放排存檔
- **列印下一個** - 列印下一個排存檔
- **傳送** - 傳送排存檔
- **移動** - 移動排存檔
- **刪除** - 刪除排存檔
- **內容** - 顯示排存檔的許多內容，並容許使用者變更其中的某些內容

當然，使用者僅能存取他們有權使用的排存檔。此外，呼叫程式能在窗格中使用 setAllowActions() 方法，阻止使用者執行動作。

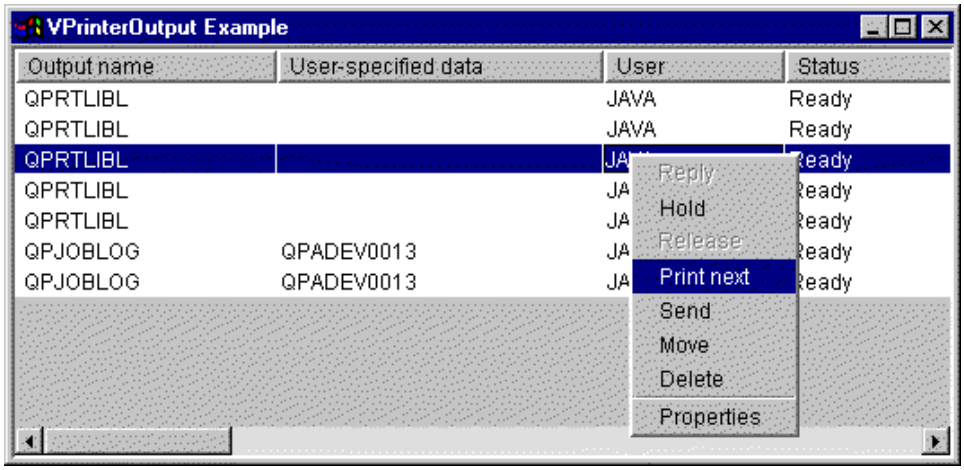
下面範例會建立 VPrinterOutput 並在 AS400ListPane 中呈現它：

```
                        // Create the VPrinterOutput object.
                        // Assume that "system" is an AS400
                        // object created and initialized
                        // elsewhere.

VPrinterOutput root = new VPrinterOutput (system);

                        // Create and load an AS400ListPane
                        // object.
     AS400ListPane listPane = new AS400ListPane (root);

listPane.load ();

                        // Add the list pane to a frame.
                        // Assume that "frame" is a JFrame
                        // created elsewhere.
     frame.getContentPane ().add (listPane);
```

範例

使用列印資源 VPrinterOutput 物件，來呈現排存檔清單。圖 1 顯示 VPrinterOutput 圖形式使用者介面元件：

圖 1：VPrinterOutput GUI 元件



SpooledFileViewer 類別:

SpooledFileViewer 類別會建立一個視窗，用來檢視已經排存要進行列印的「進階功能列印 (AFP)」及「系統網路架構」字串 (SCS) 檔案。這個類別基本上與大部分的文書處理程式一樣，會新增一個「預覽列印」功能到您的排存檔，如圖 1 中所示。

當檢視檔案佈置的精確度比列印檔案更重要時，或是當檢視資料比列印更加經濟、或是無法使用印表機時，此排存檔檢視器特別有用。

註: 主電腦伺服器上必須已安裝 SS1 選項 8 (AFP 相容字型)。

使用 SpooledFileViewer 類別

有三種建構子方法可用來建立 SpooledFileViewer 類別的實例。SpooledFileViewer() 建構子可以用來建立檢視器，但不需要相關的排存檔。如果使用這個建構子，稍後將需要使用 setSpooledFile(SpooledFile) 設定已排存的檔案。SpooledFileViewer(SpooledFile) 建構子可以用來建立特定排存檔的檢視器，並以第一頁爲起始檢視畫面。最後，SpooledFileViewer(spooledFile, int) 建構子可以用來建立特定排存檔的檢視器，並以特定頁作爲起始檢視畫面。不論使用哪一個建構子，建立好檢視器之後，就要執行 load() 呼叫，以便眞正地擷取排存檔資料。

然後，您的程式可以使用下列方法，遍訪排存檔的各個頁面：

- load FlashPage()
- load Page()
- pageBack()
- pageForward()

不過，如果您需要更詳細檢查文件的某些段落，您可以用下列方式改變每一頁的比例，來放大或縮小文件的頁影像：

- fitHeight()
- fitPage()
- fitWidth()
- actualSize()

您的程式可在呼叫 close() 方法後結束，該方法會關閉輸入串流並釋放與該串流連結的任何資源。

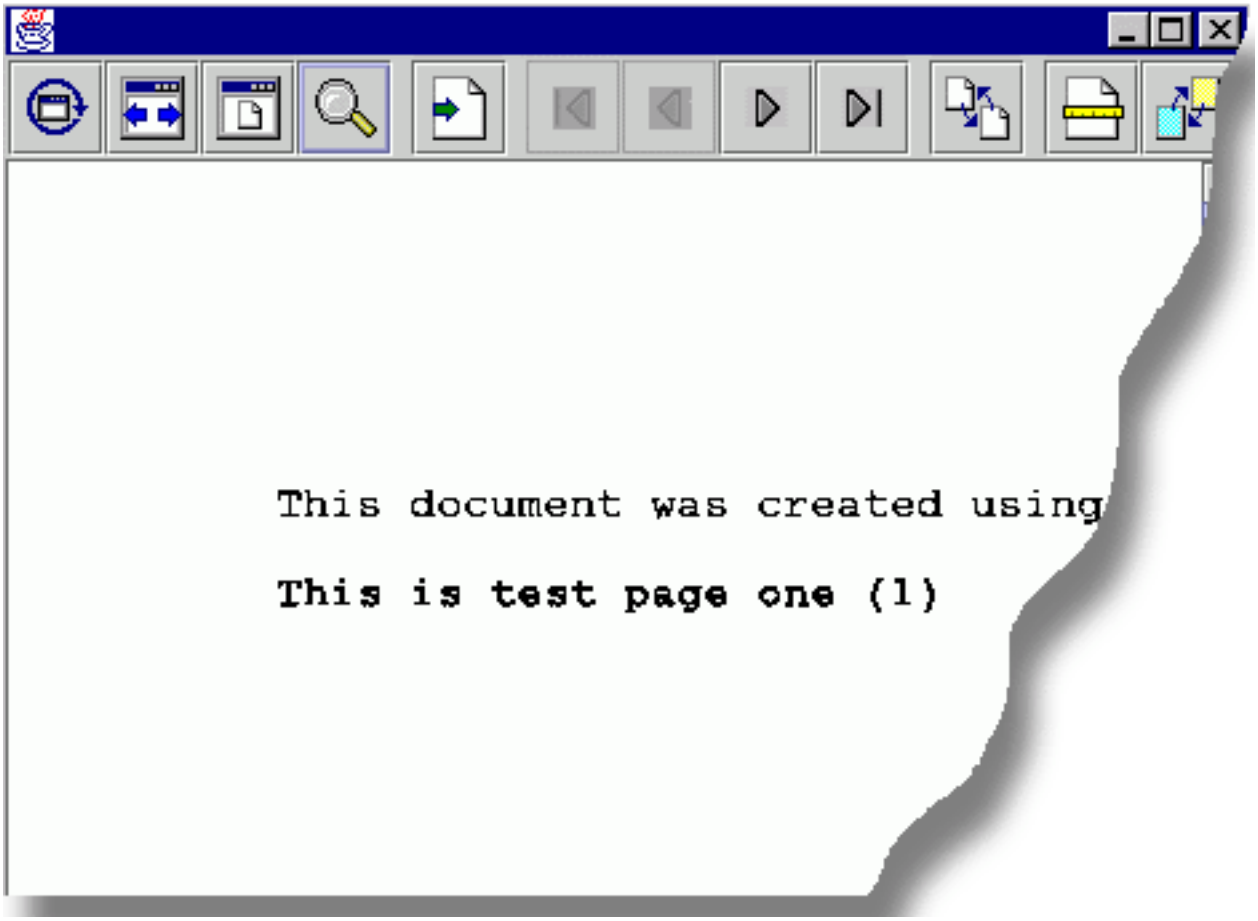
使用 SpooledFileViewer

SpooledFileViewer 類別的實例實際上是可顯示並導覽 AFP 或 SCS 排存檔之檢視器的圖形表示法。例如，下列程式可建立圖 1 中的排存檔檢視器，來顯示先前在伺服器上建立的排存檔。

註: 您可以選取圖 1 中影像上的按鈕來取得按鈕功能說明，或者 (如果您的瀏覽器未啓用 JavaScript™)，請參閱工具列說明。

```
// Assume splf is the spooled file.
// Create the spooled file viewer
SpooledFileViewer splfv = new SpooledFileViewer(splf, 1);
splfv.load();
// Add the spooled file viewer to a frame
JFrame frame = new JFrame("My Window");
frame.getContentPane().add(splfv);
frame.pack();
frame.show();
```

圖 1：SpooledFileViewer



SpooledFileViewer 工具列說明

「實際大小」按鈕會利用 actualSize() 方法，將排存檔頁影像還原成其原始的大小。

「最適寬度」按鈕會利用 fitWidth() 方法，將排存檔頁影像延伸到檢視器邊框的左右兩側。

 「最適頁面」按鈕會利用 fitPage() 方法，將排存檔頁影像上下左右延伸到佔滿整個排存檔檢視器的邊框。

 「縮放」按鈕可以讓您放大或縮小排存檔頁影像的尺寸，方法是選取預設的百分比，或在選取縮放按鈕後，在所顯現之對話框內的文字欄位中，輸入自訂的百分比。

 選取「跳至頁面」按鈕可以跳到排存檔內的特定頁面。

 選取「第一頁」按鈕會將您帶到排存檔的第一頁，並會在停止時，指出您在第一頁上。

 選取「上一頁」按鈕可將您立即帶到您目前所檢視之頁面的上一頁。

 選取「下一頁」按鈕時可將您立即帶到您目前所檢視之頁面的下一頁。

 選取「最末頁」按鈕可將您帶到排存檔的最後一頁，並在停止時，指出您在最後一頁上。

 當選取「載入快閃頁」按鈕時，它會使用 loadFlashPage() 方法載入先前所檢視的頁面。

 選取「設定紙張大小」按鈕可讓您設定紙張的尺寸。

 選取「設定檢視清晰度」按鈕可讓您設定檢視的清晰度。

Vaccess ProgramCall 類別

Vaccess 套件中的程式呼叫元件可讓 Java 程式呈現按鈕或功能表項目，以呼叫伺服器程式。您可以使用 ProgramParameter 物件來指定輸入、輸出和輸入/輸出參數。當執行程式時，輸出及輸入/輸出參數會含有伺服器程式所傳回的資料。

ProgramCallButton 物件代表一個按鈕，在按下時，會呼叫伺服器程式。ProgramCallButton 類別會延伸「Java 基礎類別 (JFC)」JButton 類別，以讓所有按鈕均有一致的外觀及行爲。

同樣地，ProgramCallMenuItem 物件代表一個功能表項目，在選取時，會呼叫伺服器程式。ProgramCallMenuItem 類別會擴充 JFC JMenuItem 類別，以便所有功能表項目也會具有一致的外觀與行爲。

若要使用 vaccess 程式呼叫元件，請同時設定系統與程式內容。經由使用建構子或透過 setSystem() 與 setProgram() 方法來設定這些特性。

下列範例會建立 ProgramCallMenuItem。在執行期間，當選取功能表選項時，它會呼叫程式：

```
                    // Create the ProgramCallMenuItem
                    // object. Assume that "system" is
                    // an AS400 object created and
                    // initialized elsewhere. The menu
                    // item text says "Select Me", and
                    // there is no icon.
ProgramCallMenuItem menuItem = new ProgramCallMenuItem ("Select Me", null, system);

                    // Create a path name object that
                    // represents program MYPROG in
                    // library MYLIB
QSYSObjectPathName programName = new QSYSObjectPathName("MYLIB", "MYPROG", "PGM");

                    // Set the name of the program.
menuItem.setProgram (programName.getPath());

                    // Add the menu item to a menu.
                    // Assume that the menu was created
                    // elsewhere.
menu.add (menuItem);
```

在執行伺服器程式時，它可能會傳回零或多則伺服器訊息。若要偵測伺服器程式的執行，請使用 addActionCompletedListener() 方法，將 ActionCompletedListener 新增到按鈕或功能表項目。每當程式執行時，它會傳送一個 ActionCompletedEvent 給所有這類接收程式。接收程式可以使用 getMessageList() 方法來擷取該程式所產生的任何伺服器訊息。

此範例會新增一個 ActionCompletedListener，來處理該程式所產生的所有伺服器訊息：

```
                    // Add an ActionCompletedListener
                    // that is implemented by using an
                    // anonymous inner class. This is a
                    // convenient way to specify simple
                    // event listeners.
menuItem.addActionCompletedListener (new ActionCompletedListener ()
{
     public void actionCompleted (ActionCompletedEvent event)
     {
                    // Cast the source of the event to a
                    // ProgramCallMenuItem.
          ProgramCallMenuItem sourceMenuItem = (ProgramCallMenuItem) event.getSource ();

                    // Get the list of server messages
                    // that the program generated.
          AS400Message[] messageList = sourceMenuItem.getMessageList ();

                    // ... Process the message list.
     }
});
```

參數

您可以使用 ProgramParameter 物件，在 Java 程式與伺服器程式之間傳遞參數資料。輸入資料是使用 setInputData() 方法來設定的。在程式執行後，可以使用 getOutputData() 方法來擷取輸出資料。

每一個參數都是一個位元組陣列。Java 程式負責在 Java 與伺服器格式之間轉換位元組陣列。資料轉換類別可提供轉換資料的方法。

您可以使用 addParameter() 方法，一次將一個參數新增到程式呼叫圖形使用者介面元件中，或是使用 setParameterList() 方法，一次將所有參數新增到程式呼叫圖形使用者介面元件中。

關於如何使用 ProgramParameter 物件的詳細資訊，請參閱 ProgramCall 存取類別。

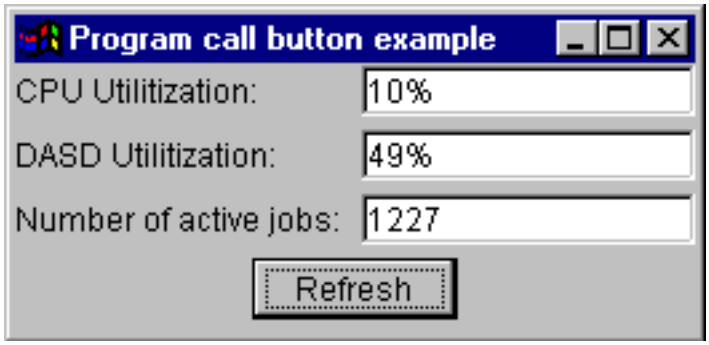
下列範例會新增兩個參數：

```
                        // The first parameter is a String
                        // name of up to 100 characters.
                        // This is an input parameter.
                        // Assume that "name" is a String
                        // created and initialized elsewhere.

AS400Text parm1Converter = new AS400Text (100, system.getCcsid (),
system);
     ProgramParameter parm1 = new ProgramParameter (parm1Converter.toBytes (name));
     menuItem.addParameter (parm1);

                        // The second parameter is an Integer
                        // output parameter.
     AS400Bin4 parm2Converter = new AS400Bin4 ();

ProgramParameter parm2 = new ProgramParameter
(parm2Converter.getByteLength ());
     menuItem.addParameter (parm2);

                        // ... after the program is called,
                        // get the value returned as the
                        // second parameter.
     int result = parm2Converter.toInt (parm2.getOutputData ());
```

範例

在應用程式中使用 ProgramCallButton 的範例。圖 1 顯示 ProgramCallButton 的外觀：

圖 1：在應用程式中使用 ProgramCallButton



Vaccess 記錄層次存取類別

Vaccess 套件中的記錄層次存取類別可讓 Java 程式以各種檢視畫面來呈現伺服器檔案。

可用的元件如下：

- RecordListFormPane 以套表來呈現伺服器檔案的記錄清單。
- RecordListTablePane 以表格來呈現伺服器檔案的記錄清單。
- RecordListTableModel 以表格管理伺服器檔案的記錄清單。

索引存取

您可以使用伺服器檔案使用記錄層次存取圖形式使用者介面元件以及索引存取。索引存取表示 Java 程式可藉由指定索引來存取檔案的記錄。

索引存取與每一個記錄層次存取圖形式使用者介面元件的運作方式是相同的。請使用 setKeyed() 來指定索引存取，而非循序存取。使用建構子或 setKey() 方法來指定索引鍵。請參閱指定索引鍵，取得金鑰的指定方法詳細資訊。

依據預設值，僅有其索引等於指定的索引的記錄才會顯示。若要變更，請使用建構子或 setSearchType() 方法，指定 searchType 特性。可能的選擇如下：

- KEY_EQ - 顯示其索引等於指定的索引的記錄。
- KEY_GE - 顯示其索引大於或等於指定的索引的記錄。
- KEY_GT - 顯示其索引大於指定的索引的記錄。
- KEY_LE - 顯示其索引小於或等於指定的索引的記錄。
- KEY_LT - 顯示其索引小於指定的索引的記錄。

下面範例會建立一個 RecordListTablePane 物件，來顯示小於或等於某個索引的所有記錄。

```
                    // Create a key that contains a
                    // single element, the Integer 5.
Object[] key = new Object[1];
key[0] = new Integer (5);

                    // Create a RecordListTablePane
                    // object. Assume that "system" is an
                    // AS400 object that is created and
                    // initialized elsewhere. Specify
                    // the key and search type.
RecordListTablePane tablePane = new RecordListTablePane (system,
     "/QSYS.LIB/QGPL.LIB/PARTS.FILE", key, RecordListTablePane.KEY_LE);

                    // Load the file contents.
tablePane.load ();

                    // Add the table pane to a frame.
                    // Assume that "frame" is a JFrame
                    // created elsewhere.
frame.getContentPane ().add (tablePane);
```

RecordListFormPane 類別:

RecordListFormPane 以套表呈現伺服器檔案內容。表單會一次顯示一筆記錄，並提供一些按鈕，容許使用者向前捲動、向後捲動到第一筆或最後一筆記錄，或是復新檔案內容的檢視畫面。

若要使用 RecordListFormPane，請設定系統與 fileName 特性。使用建構子或 setSystem() 和 setFileName() 方法來設定這些內容。使用 load() 來擷取檔案內容並顯示第一筆記錄。當檔案內容不再需要時，請呼叫 close()，以確保結束檔案。

下面範例會建立 RecordListFormPane 物件，並將它新增到頁框中：

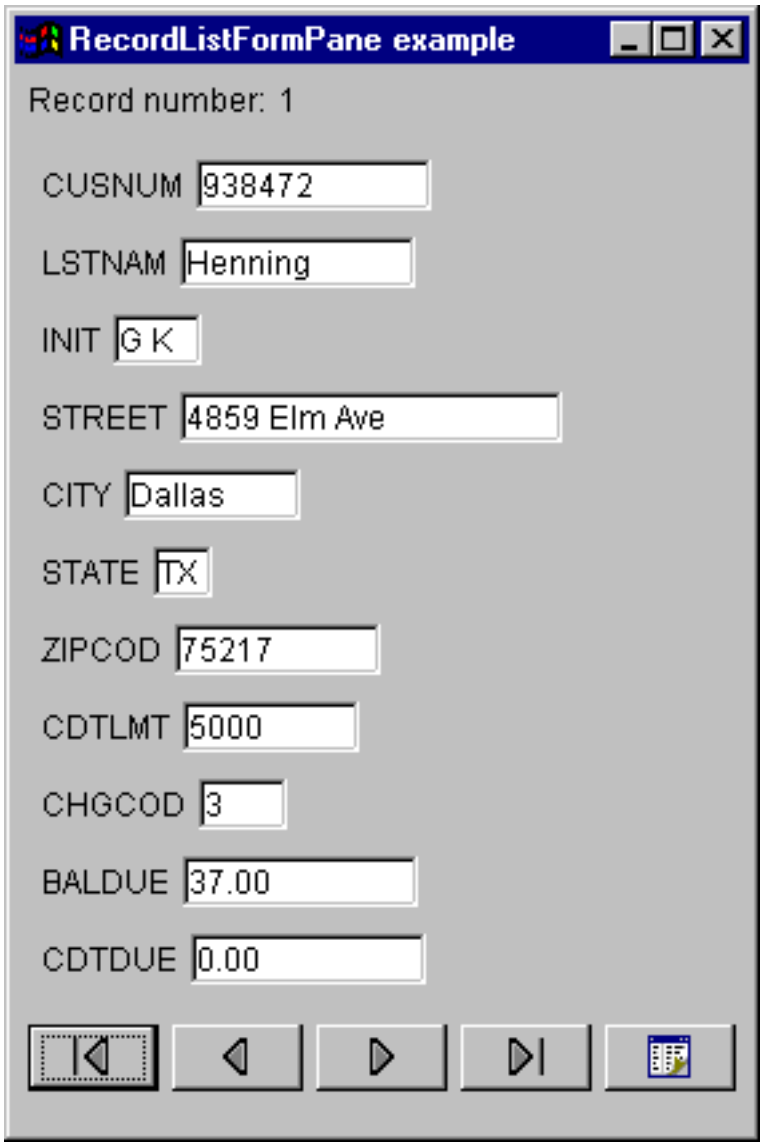
```
                    // Create a RecordListFormPane
                    // object. Assume that "system" is
                    // an AS400 object that is created
                    // and initialized elsewhere.
RecordListFormPane formPane = new RecordListFormPane (system,
                              "/QSYS.LIB/QIWS.LIB/QCUSTCDT.FILE");

                    // Load the file contents.
formPane.load ();
```

```
                    // Add the form pane to a frame.
                    // Assume that "frame" is a JFrame
                    // created elsewhere.
    frame.getContentPane ().add (formPane);
```

範例

呈現會顯示檔案內容的 RecordListFormPane。圖 1 顯示 RecordListFormPane 圖形式使用者介面元件：

圖 1：RecordListFormPane GUI 元件



RecordListTablePane 類別:

RecordListTablePane 以表格呈現伺服器檔案內容。表格中的每一列顯示來自檔案的記錄，而每一欄則顯示欄位。

若要使用 RecordListTablePane，請設定系統與 fileName 特性。使用建構子或 setSystem() 和 setFileName() 方法來設定這些內容。使用 load() 來擷取檔案內容，並在表格中呈現記錄。當檔案內容不再需要時，請呼叫 close()，以確保結束檔案。

下面範例會建立 RecordListTablePane 物件，並將它新增到頁框中：

```
                    // Create an RecordListTablePane
                    // object. Assume that "system" is
                    // an AS400 object that is created
                    // and initialized elsewhere.
    RecordListTablePane tablePane = new RecordListTablePane (system,
                                "/QSYS.LIB/QIWS.LIB/QCUSTCDT.FILE");
                    // Load the file contents.
    tablePane.load ();

                    // Add the table pane to a frame.
                    // Assume that "frame" is a JFrame
                    // created elsewhere.
    frame.getContentPane ().add (tablePane);
```

RecordListTablePane 與 RecordListTableModel 類別:

RecordListTablePane 是使用模型-概略表-控制器範例來實施，在此資料與使用者介面被分成不同的類別。此實作方式可整合 SQLResultSetTableModel 與「Java 基礎類別 (JFC)」的 JTable。RecordListTableModel 類別會擷取並管理檔案內容，而 JTable 則會以圖形方式顯示檔案內容並處理使用者互動。

RecordListTablePane 會提供足夠的功能來符合大部分的需求。不過，如果呼叫程式需要更進一步控制 JFC 元件，則呼叫程式可以直接使用 RecordListTableModel，並提供與不同圖形使用者介面元件的自訂整合。

若要使用 RecordListTableModel，請設定系統與 fileName 特性。使用建構子或 setSystem() 和 setFileName() 方法來設定這些內容。使用 load() 來擷取檔案內容。當檔案內容不再需要時，請呼叫 close()，以確保結束檔案。

下面範例會建立 RecordListTableModel 物件，並以 JTable 呈現它：

```
                // Create a RecordListTableModel
                // object. Assume that "system" is
                // an AS400 object that is created
                // and initialized elsewhere.
RecordListTableModel tableModel = new RecordListTableModel (system, "/QSYS.LIB/QIWS.LIB/QCUSTCDT.FILE");

                // Load the file contents.
tableModel.load ();

                // Create a JTable for the model.
JTable table = new JTable (tableModel);

                // Add the table to a frame.  Assume
                // that "frame" is a JFrame
                // created elsewhere.
frame.getContentPane ().add (table);
```

ResourceListPane 與 ResourceListDetailsPane

使用 ResourceListPane 和 ResourceListDetailsPane 類別，在圖形式使用者介面 (GUI) 中呈現資源清單。

- ResourceListPane 會在圖形式的 javax.swing.JList 中顯示資源清單的內容。清單中所顯示的每個項目代表資源清單中的一個資源物件。
- ResourceListDetailsPane 會在圖形式的 javax.swing.JTable 中顯示資源清單的內容。表格中所顯示的每一列代表資源清單中的一個資源物件。

ResourceListDetailsPane 的表格直欄將指定爲直欄屬性 ID 的陣列。每一個陣列的元素是表格中的一個直欄，而每一個資源物件是表格中的一列。

在預設的情況下，ResourceListPane 和 ResourceListDetailsPane 二者都會啓用蹦現功能表。

大部分的錯誤將報告爲 com.ibm.as400.vaccess.ErrorEvents，而非丟出異常。因此，請接收 ErrorEvents，來診斷錯誤狀況，並從錯誤狀況中回復。

範例：在 GUI 中顯示資源清單

此範例會針對系統上的所有使用者建立一個 ResourceList，並將其顯示在 GUI 中 (明細窗格)：

```
    // Create the resource list.
    AS400 system = new AS400("MYSYSTEM", "MYUSERID", "MYPASSWORD");
    RUserList userList = new RUserList(system);

    // Create the ResourceListDetailsPane.In this example,
 // there are two columns in the table.The first column
 // contains the icons and names for each user.The
 // second column contains the text description for each
    // user.
    Object[] columnAttributeIDs = new Object[] { null, RUser.TEXT_DESCRIPTION };
    ResourceListDetailsPane detailsPane = new ResourceListDetailsPane();
    detailsPane.setResourceList(userList);
    detailsPane.setColumnAttributeIDs(columnAttributeIDs);

    // Add the ResourceListDetailsPane to a JFrame and show it.
   JFrame frame = new JFrame("My Window");
    frame.getContentPane ().add (detailsPane);
   frame.pack();
   frame.show();

    // The ResourceListDetailsPane will appear empty until
    // we load it.This gives us control of when the list
 // of users is retrieved from the iSeries.

detailsPane.load ();
```

系統狀態類別

vaccess 套件中的「系統狀態」元件可讓您使用現有的 AS400Panes，來建立 GUI。您也可以選擇使用「Java 基礎類別 (JFC)」，建立自己的 GUI。VSystemStatus 物件代表伺服器上的系統狀態。VSystemPool 物件代表伺服器上的系統儲存區。 VSystemStatusPane 代表顯示系統狀態資訊的視覺化窗格。

VSystemStatus 類別可讓您在 GUI 環境內，取得伺服器階段作業狀態的相關資訊：

- getSystem() 方法將傳回包含系統狀態資訊的伺服器
- getText() 方法將傳回說明文字
- setSystem() 方法將設定系統狀態資訊所在的伺服器

除了上面提到的方法外，您也可以在 GUI 中存取及變更系統儲存區資訊。

您可以將 VSystemStatus 與 VSystemStatusPane 搭配使用。VSystemPane 是顯示系統狀態及系統儲存區資訊的視效窗格。

VSystemPool 類別:

VSystemPool 類別可讓您使用 GUI 設計，從伺服器擷取及設定系統儲存區資訊。VSystemPool 可和 vaccess 套件中的各種窗格配合使用，包括 VSystemStatusPane。

下列清單是可在 VSystemPool 中使用的部分方法：

- getActions() 方法將傳回您可以執行的動作清單
- getSystem() 方法將傳回找到系統儲存區資訊的伺服器

- setSystemPool() 方法將設定系統儲存區物件

VSystemStatusPane 類別:

VSystemStatusPane 類別可讓 Java 程式顯示系統狀態及系統儲存區資訊。

VSystemStatusPane 包括下列方法：

- getVSystemStatus()：將 VSystemStatus 資訊傳回 VSystemStatusPane 中。
- setAllowModifyAllPools()：設定該值，決定是否可以修改系統儲存區資訊。

下面範例將告訴您如何使用 VSystemStatusPane 類別：

```
// Create an as400 object.
AS400 mySystem = new AS400("mySystem.myCompany.com");

// Create a VSystemStatusPane
VSystemStatusPane myPane = new VSystemStatusPane(mySystem);

// Set the value to allow pools to be modified
myPane.setAllowModifyAllPools(true);

//Load the information
myPane.load();
```

系統值 GUI

Vaccess 套件中的系統值元件可讓 Java 程式使用現有的 AS400Panes，或使用「Java 基礎類別 (JFC)」建立您自己的窗格，藉以建立 GUI。VSystemValueList 物件代表伺服器上的系統值清單。

若要使用系統值 GUI 元件，請使用建構子或透過 setSystem() 方法，來設定系統名稱。

範例 下列範例使用 AS400Explorer 窗格來建立一個系統值 GUI：

```
//Create an AS400 object
AS400 mySystem = newAS400("mySystem.myCompany.com");
VSystemValueList mySystemValueList = new VSystemValueList(mySystem);
as400Panel=new AS400ExplorerPane((VNode)mySystemValueList);
//Create and load an AS400ExplorerPane object
as400Panel.load();
```

Vaccess 使用者與群組類別

vaccess 套件中的使用者與群組元件可讓您透過 VUser 類別，來呈現伺服器使用者與群組清單。

可用的元件如下：

- AS400Panes 是一些 GUI 元件，它們會呈現及容許操作一個或多個伺服器資源。
- VUserList 物件是一種資源，它代表要在 AS400Panes 中使用的伺服器使用者與群組的清單。
- VUserAndGroup 物件是要在 AS400Panes 中使用的資源，用來代表伺服器使用者的群組。它可讓 Java 程式列出所有使用者，列出所有群組或列出不在群組中的使用者。

AS400Pane 與 VUserList 物件可以一起使用，以呈現清單的多種檢視方式。它們也可以用來容許使用者選取使用者和群組。

若要使用 VUserList，首先您必須設定系統特性。此內容可以使用建構子或透過 setSystem() 方法來設定。接著使用 AS400Pane 的建構子或 setRoot() 方法， 將 VUserList 物件「插入」AS400Pane 中作爲根。

VUserList 含有其它部分有用的內容，可用來定義 AS400Panes 中所呈現的使用者與群組集：

- 使用 setUserInfo() 方法來指定應該出現的使用者類型。
- 使用 setGroupInfo() 方法來指定群組名稱。

您可以使用 VUserAndGroup 物件，來取得系統上的「使用者與群組」相關資訊。在您能取得特定物件的相關資訊之前，您需要載入該資訊，才能存取它。您可以使用 getSystem 方法，來顯示能在其中找到該資訊的伺服器。

當建立 AS400Pane 物件與 VUserList 或 VUserAndGroup 物件時，它們會起始設定爲預設狀態。尚未載入使用者和群組的清單。若要載入內容，Java 程式必須在任一物件中明確地呼叫 load() 方法，來起始與伺服器的通訊，以便收集清單的內容。

在執行時間，以滑鼠右鍵按一下使用者、使用者清單或群組以顯示捷徑功能表。從捷徑功能表中選取**內容**，針對選取的物件執行動作：

- 使用者 - 顯示使用者資訊的清單，包括說明、使用者類別、狀態、工作說明、輸出資訊、訊息資訊、國際資訊、安全資訊及群組資訊。
- 使用者清單 - 處理使用者資訊和群組資訊內容。您也可以變更清單的內容。
- 使用者與群組 - 顯示內容，例如使用者名稱及說明。

使用者僅能存取他們有權使用的使用者和群組。此外，Java 程式可在窗格中使用 setAllowActions() 方法，以防止使用者執行動作。

下面範例會建立 VUserList 並在 AS400DetailsPane 中呈現它：

```
    // Create the VUserList object.
    // Assume that "system" is an AS400
    // object created and initialized
    // elsewhere.
    VUserList root = new VUserList (system);

    // Create and load an
    // AS400DetailsPane object.
    AS400DetailsPane detailsPane = new AS400DetailsPane (root);

detailsPane.load ();

    // Add the details pane to a frame.
    // Assume that "frame" is a JFrame
    // created elsewhere.
    frame.getContentPane ().add (detailsPane);
```

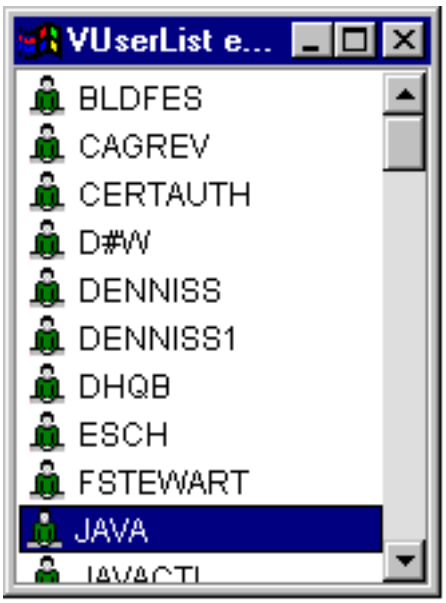
下列的範例顯示如何使用 VUserAndGroup 物件：

```
    // Create the VUserAndGroup object.
    // Assume that "system" is an AS400 object created and initialized elsewhere.
    VUserAndGroup root = new VUserAndGroup(system);

    // Create and Load an AS400ExplorerPane
    AS400ExplorerPane explorerPane = new AS400ExplorerPane(root);
    explorerPane.load();

    // Add the explorer pane to a frame
    // Assume that "frame" is a JFrame created elsewhere
    frame.getContentPane().add(explorerPane);
```

其他範例

使用 AS400ListPane 搭配 VUserList 物件，呈現系統中的使用者清單。

下面影像顯示 VUserList 圖形使用者介面元件：



Graphical Toolbox 及 PDML

Graphical Toolbox 是一套 UI 工具集，能讓您輕鬆地使用 Java 建立自訂的使用者介面畫面。

您可以把畫面納入 Java 應用程式、Applet 或 「iSeries 領航員」外掛程式。畫面可能包含由 iSeries 所取得的資料，或由其他來源取得的資料，如本端檔案系統中的檔案，或網路上的程式。

GUI Builder 是 WYSIWYG 視覺化編輯器，可用來建立 Java 對話框、內容表及精靈。利用 GUI Builder，您可以新增、排列或編輯畫面中的使用者介面控制項，然後預覽畫面以驗證佈置結果是否如您預期。您所建立的畫面定義可以用於對話框中、插入內容表及精靈中，或放入分割窗格、重疊窗格及標籤窗格。GUI Builder 也可讓您建置功能表列、工具列及上下文功能表定義。您還可以把 JavaHelp 納入畫面中，包括上下文相關的說明在內。

Resource Script Converter 會將 Windows 資源 scripts 轉換爲 Java 程式可以使用的 XML 表示法。利用 Resource Script Converter，您可以在您現有的 Windows 對話框及功能表中處理 Windows 資源 scripts (RC 檔)。然後，這些已轉換的檔案可以使用 GUI Builder 加以編輯。使用 Resource Script Converter 及 GUI Builder，可以從 RC 檔製作內容表及精靈。

這兩種工具背後的基礎在於名爲 **Panel Definition Markup Language** 或 **PDML** 的新技術。PDML 是基於 Extensible Markup Language (XML)，它定義出說明使用者介面元素佈置方式用的獨立式平台語言。只要於 PDML 中定義您的畫面，您就可以使用由 Graphical Toolbox 所提供的 執行時間 API 來顯示畫面。API 會解譯 PDML，並使用「Java 基礎類別」來呈現您的使用者介面，藉以顯示您的畫面。

註: 若要使用 PDML，您必須執行 Java Runtime Environment 1.4 版或更新版本。

Graphical Toolbox 的好處

撰寫的程式較少，並可節省時間

使用 Graphical Toolbox，您就可以輕鬆又快速地建立 Java 使用者介面。GUI Builder 可讓您精確地控制畫面中 UI 元素的佈置情形。因爲佈置是以 PDML 來說明，所以您不需要爲了定義使用者介面而開發任何 Java 程式碼，也不需要爲了進行變更而重新編譯程式碼。因此，只需要以相當少的時間，即可建立及維護您的 Java 應用程式。Resource Script Converter 可讓您迅速且輕易地將大量的 Windows 畫面移轉成 Java。

自訂解說

於 PDML 中定義使用者介面可帶來其它好處。因爲一張畫面的所有資訊都合併成正式的標記語言，所以工具可以強化爲代替程式開發者來執行更多的服務。例如，GUI Builder 和 Resource Script Converter 都能 夠產生畫面線上說明的 HTML 基本架構。由您決定需要哪些解說主題，解說主題就會自動按照您的要求建立起來。解說主題的錨點標籤會直接建置於說明的基本架構當中，此有助於撰寫者專注於開發適當的內容。Graphical Toolbox 執行時間環境會回應使用者的要求，自動顯示出正確的解說主題。

畫面到程式的自動整合

此外 PDML 還能提供一些標記，使畫面中的每一個控制項與 JavaBean 中的屬性產生關聯。一旦您已識別出會提供資料給畫面的 Bean 類別，並且也將每個適當的控制項與屬性相結合，便可要求工具產生 Bean 物件的 Java 原始程式碼架構。到了執行時間，「圖形化工具箱」會自動在 bean 及 您所識別的畫面控制項之間轉送資料。

平台獨立

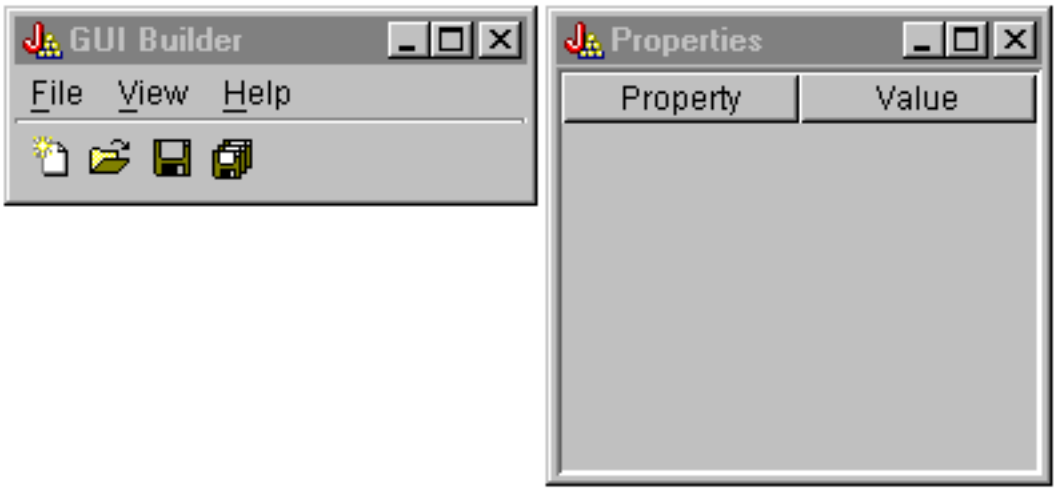
Graphical Toolbox 執行時間環境可提供事件處理的支援、使用者資料驗證，以及畫面元素之間的一般互動類型。您的使用者介面之正確平台外觀及風格將自動依據基本的作業系統來設定，而且 GUI Builder 可讓您在外觀及風格之間切換，以便讓您在不同的平台中評估畫面的外觀效果。

Graphical Toolbox 提供您兩種工具，因此，有兩種自動建立使用者介面的方法。您可以使用 GUI Builder 迅速且輕易地從頭開始建立新畫面，或您可以使用 Resource Script Converter，將現有的 Windows 型的畫面轉換爲 Java。然後可以使用 GUI Builder 編輯已轉換的檔案。這兩種工具都支援國際化。

GUI Builder

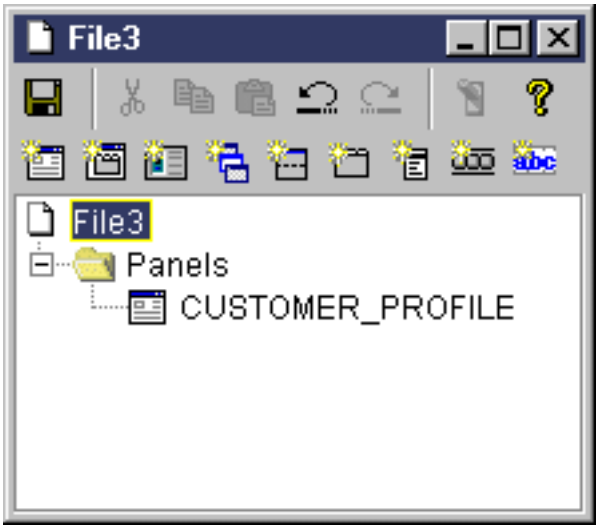
當您第一次呼叫 GUI Builder 時，會顯示兩個視窗，如圖 1 所示：

圖 1： GUI Builder 視窗



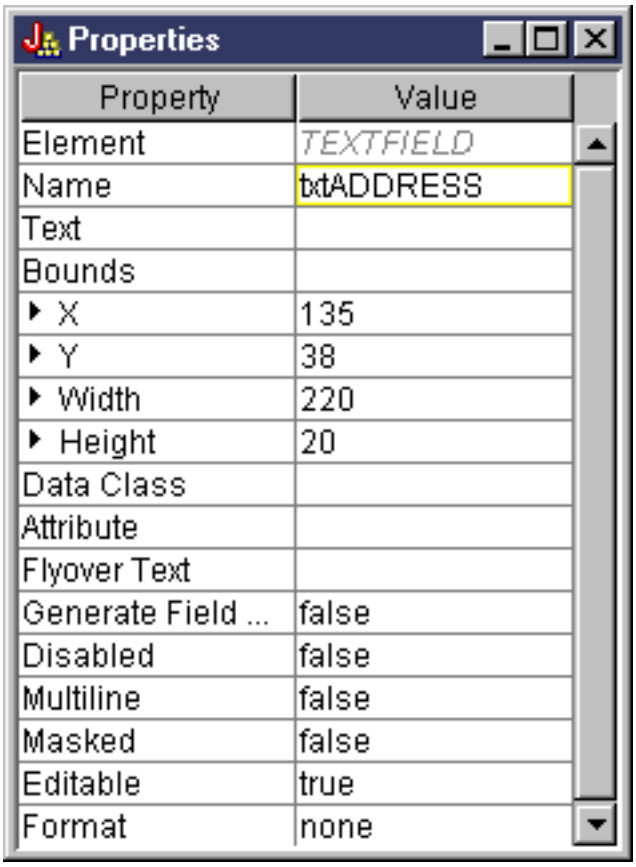
請使用 File Builder 視窗，建立及編輯您的 PDML 檔。

圖 2：File Builder 視窗



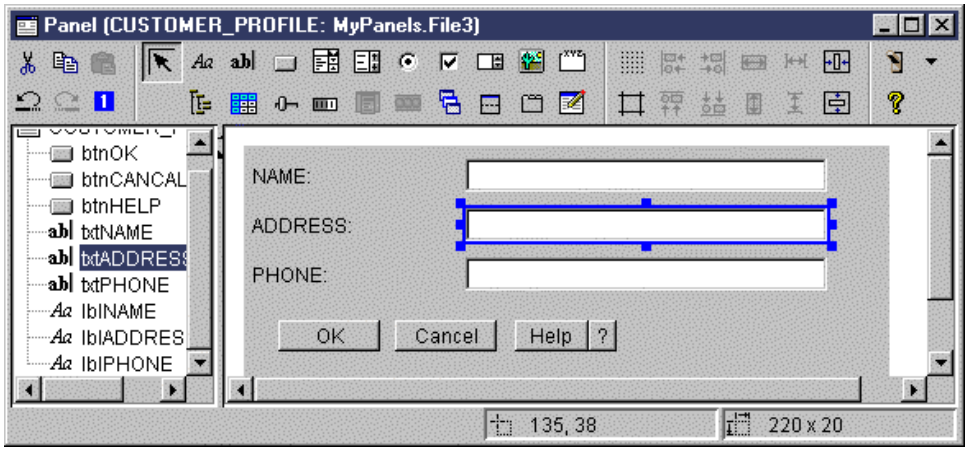
請使用 Properties 視窗，來檢視或變更目前選取的控制項內容。

圖 3：Properties 視窗



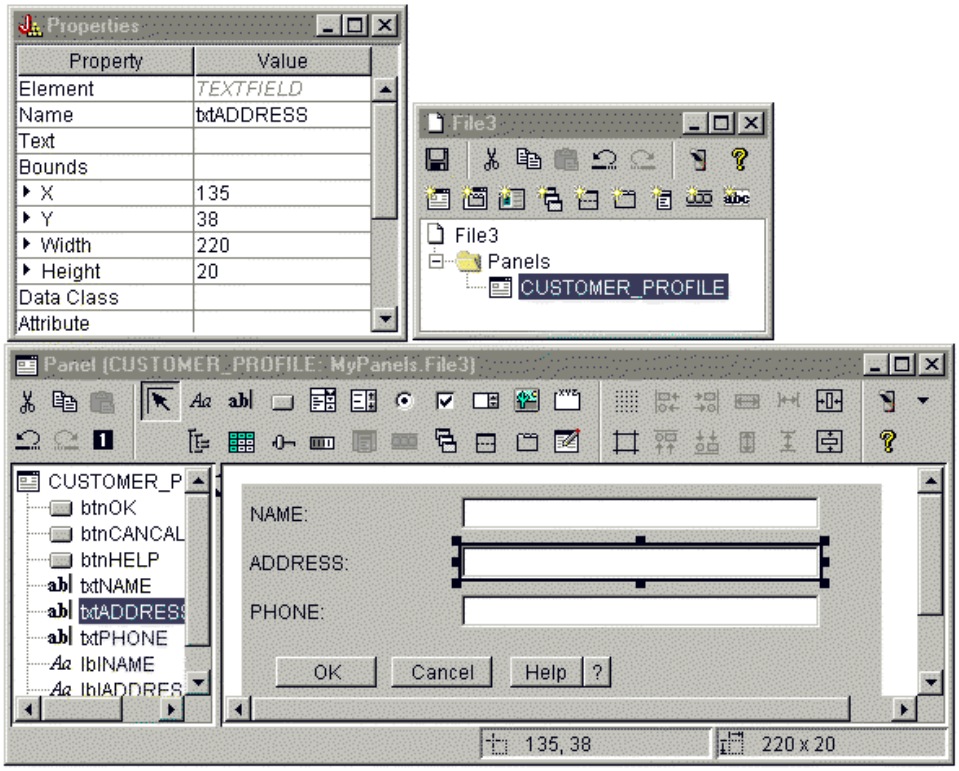
請使用 Panel Builder 視窗，建立及編輯您的圖形式使用者介面元件。從工具列中選取想要的元件，然後按一下畫面，將它放在您要的位置上。工具列也提供了一些機能，以對齊控制項群組、預覽畫面及要求 GUI Builder 功能的線上說明。 請參閱 GUI Builder Panel Builder 工具列的說明以取得每一個圖示功能的說明。

圖 4：Panel Builder 視窗



在 Panel Builder 視窗中會顯示要編輯的畫面。 圖 5 顯示各視窗如何一起作用：

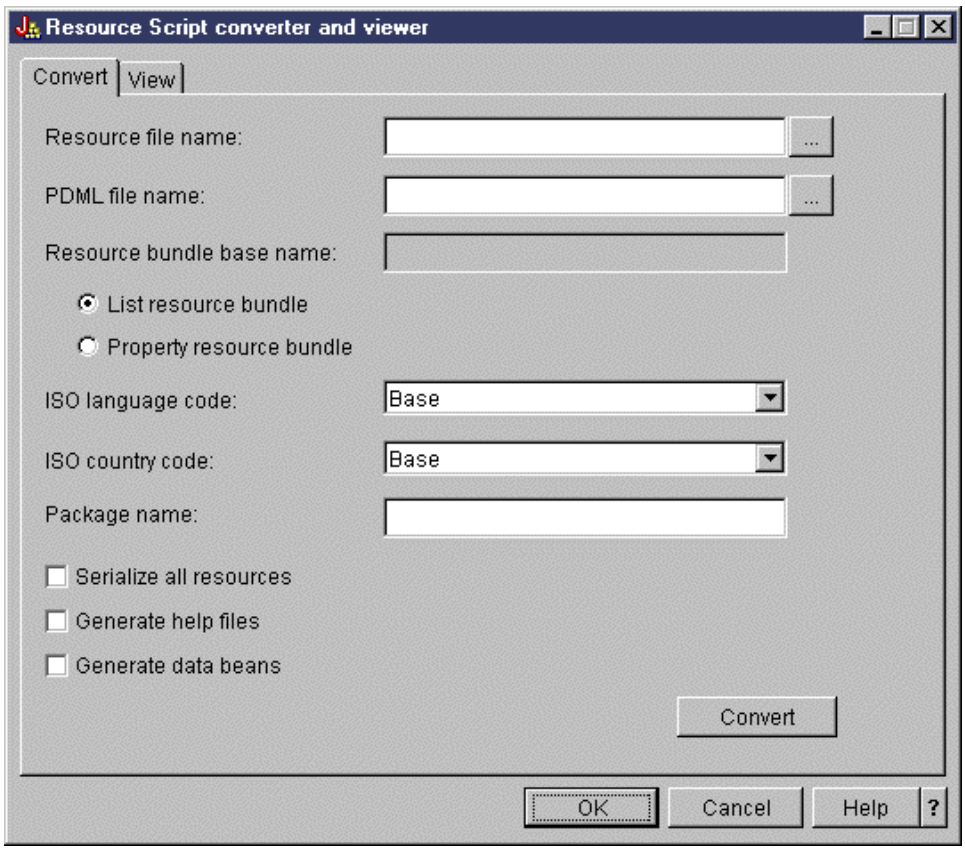
圖 5：GUI Builder 視窗一起作用的範例



Resource Script Converter

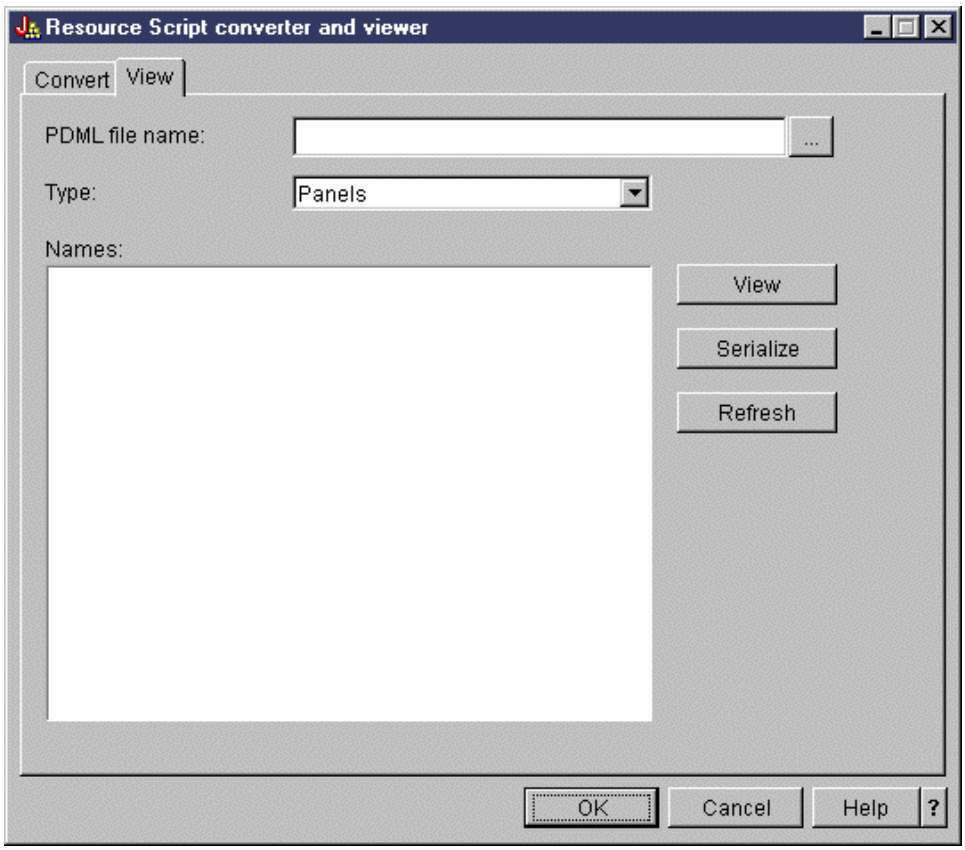
Resource Script Converter 由含標籤的兩個窗格對話框所組成。於**轉換**窗格上，您可以指定要轉換成 PDML 之 Windows RC 檔案的 Microsoft 或 VisualAge® 名稱。您可以指定目標 PDML 檔的名稱，及將含有畫面之已轉換字串的相關 Java 資源軟體組。此外，您可以要求爲畫面產生線上說明架構，產生提供資料給畫面之物件的 Java 原始程式碼架構，以及將畫面定義序列化，以改善執行時間的效能。Converter 的線上說明提供 Conver 窗格上每一輸入欄位的詳細說明。

圖 6：Resource Script Converter 的 Convert 窗格



順利轉換之後，您可以使用**檢視**窗格來檢視新建立之 PDML 檔的內容，以及預覽新的 Java 畫面。如果需要，您可使用 GUI Builder 來稍微調整 畫面。在執行轉換之前，Converter 一律會檢查現存的 PDML 檔，並嘗試保留任何變更，以便您在稍後執行轉換時可能需要它。

圖 7：Resource Script Converter 檢視窗格



設置 Graphical Toolbox

Graphical Toolbox 是以一組 JAR 檔來傳遞。 如欲設定 Graphical Toolbox，您必須在您的工作站上安裝 JAR 檔，並設定您的 CLASSPATH 環境變數。

您也必須確定您的工作站符合執行 IBM Toolbox for Java 的基本要求。

在您的工作站上安裝 Graphical Toolbox

若要使用 Graphical Toolbox 開發 Java 程式，請先在您的工作站上安裝 Graphical Toolbox JAR 檔。方法有下列幾種：

轉送 JAR 檔

附註：以下的清單代表轉送 JAR 檔的幾種不同方法。您的 iSeries 上必須安裝 IBM Toolbox for Java 授權程式。此外，您也必須從 Sun JavaHelp 網站 下載 JavaHelp 的 JAR 檔 jhall.jar。

- 使用 FTP (確定您是以二進位模式轉送檔案)，並從目錄 **/QIBM/ProdData/HTTP/Public/jt400/lib** 複製 JAR 檔到您工作站上的本端目錄中
- 使用 iSeries Access for Windows 來對映網路磁碟機。

使用 iSeries Access for Windows 安裝 JAR 檔

您也可以在安裝 iSeries Access for Windows 的時候安裝 Graphical Toolbox。IBM Toolbox for Java 現已隨 iSeries Access for Windows 一起提供。如果您是首次安裝 iSeries Access for Windows，請先選擇「自訂安裝」，再選取安裝功能表上的 **IBM Toolbox for Java** 元件。如果您已安裝了 iSeries Access for Windows，則可以使用「選擇性安裝」程式來安裝這個元件 (若它不存在的話)。

設定 classpath

若要使用 Graphical Toolbox，您必須新增這些 JAR 檔到您 的 CLASSPATH 環境變數中 (或在指令行上的 `classpath` 選項中指定它們)。

例如，如果您已經把這些檔案複製到工作站的 **C:\gtbox\lib** 目錄下，則必須新增下列路徑名稱到 classpath：

```
C:\gtbox\lib\uitools.jar;
     C:\gtbox\lib\jui400.jar;
     C:\gtbox\lib\data400.jar;
     C:\gtbox\lib\util400.jar;
     C:\gtbox\lib\jhall.jar;
```

同時，您也必須將 XML 剖析器加入 CLASSPATH。如需詳細資訊，請參閱下列網頁：

第 370 頁的『XML 剖析器與 XSLT 處理器』

如果您已使用 iSeries Access for Windows 安裝了 Graphical Toolbox，則 JAR 檔 (jhall.jar 除外) 將位於 iSeries Access for Windows 所安裝之磁碟機的 **\Program Files\Ibm\Client Access\jt400\lib** 目錄中。iSeries Access for Windows 會將 jhall.jar 安裝在 **\Program Files\Ibm\Client Access\jre\lib** 目錄中。您 classpath 中的路徑名稱對此會有所反映。

JAR 檔案說明

- **uitools.jar**： 含有 GUI Builder 與「資源 Script 轉換器」工具。
- **jui400.jar**： 含有 Graphical Toolbox 的執行時間 API。Java 程式會使用這個 API，顯示使用工具所建構的畫面。這些類別可隨著應用程式一起重新分送。
- **data400.jar**： 含有「程式呼叫標記語言 (PCML)」的執行時間 API。Java 程式使用此 API 來呼叫其參數及回覆值是使用 PCML 來識別的 iSeries 程式。這些類別可隨著應用程式一起重新分送。
- **util400.jar**：含有格式化 iSeries 資料及處理 iSeries 訊息所需的公用程式類別。這些類別可隨著應用程式一起重新分送。
- **jhall.jar**： 含有 JavaHelp 類別，這些類別可爲您使用 GUI Builder 所建置成的畫面，顯示出線上說明和環境定義相關的說明。
- **XML 剖析器**： 含有 API 類別用來解譯 PDML 與 PCML 文件的 XML 剖析器。

註: 您可以使用國際化版本的 GUI Builder 與「資源 Script 轉換器」工具。若要執行非美式英文的版本，您必須在 Graphical Toolbox 安裝作業中新增適合您語言及國家或地區所使用的正確版本的 **uitools.jar**。iSeries 伺服器的 **/QIBM/ProdData/HTTP/Public/jt400/Mri29xx** 提供了這些 JAR 檔，其中 29xx 就是與您的語言及國家或地區對應的 4 位數 i5/OS NLV 碼。(各種 Mri29xx 目錄中的 JAR 檔名包含代表 Java 語言碼及國家或地區碼的 2 字元字尾)。這個額外的 JAR 檔將會新增到您的 classpath 中，並且搜尋次序在 **uitools.jar** 之前。

使用 Graphical Toolbox

一旦安裝了 Graphical Toolbox 後，請遵循這些鏈接來學習如何使用工具：

- 使用 GUI 建置器
- 使用 Resource Script Converter

建立您的使用者介面

您可以使用 GUI Builder 工具建立使用者介面。

若要啓動 GUI Builder，請使用下列指令：

```
java com.ibm.as400.ui.tools.GUIBuilder [-plaf look and feel]
```

如果您未設定 CLASSPATH 環境變數來包含 Graphical Toolbox JAR 檔，則您須要在指令行上，使用 `classpath` 選項來指定它們。 請參閱設置 Graphical Toolbox。

選項 `-plaf` *外觀與操作方式*

您屬意的平台外觀與操作方式。此選項可讓您置換您依照目前開發的平台所設定的外觀與操作方式預設值，因此您可以預覽畫面，察看它們在不同的作業系統平台上的外觀。 系統可接受下列外觀與操作方式的值：

- Windows
- Metal
- Motif

目前，Swing 1.1 可以支援的其它外觀與感覺的屬性並不爲 GUI Builder 所支援。

使用者介面資源的類型

當您第一次啓動 GUI Builder 時，必須建立新的 PDML 檔。從 GUI Builder 視窗的功能表列上，選取 **File --> New File**。 建立新的 PDML 檔案之後，您就可以隨意定義下列型類的 UI 資源包含在內。

Panel (畫面)

基本資源類型。描述一個矩形區，UI 元素就在這個區域內排列。UI 元素可以由簡式控制項所組成，如圓鈕或文字欄位、影像、動畫、自訂控制項或更複雜的次畫面 (請參閱底下的分割窗格、疊窗格及標籤窗格)。一個畫面可以定義獨立式視窗或對話框的佈置，或者也可以定義在另一個 UI 資源中包含的次畫面之一。

Menu (功能表)

蹦現視窗含有一或多個可選取的動作，每一個都由一個字串代表 (如 Cut、Copy 及 Paste)。您可以定義每一個動作的助記符號及快速鍵、插入分隔字元及階式排列子功能表，或定義特殊的勾選或圓鈕功能表項目。功能表資源可以當作獨立式上下文功能表、功能表列中的下拉式功能表使用，或它可以自己定義與畫面資源相關的功能表列。

Toolbar (工具列)

視窗是由一系列的按鈕組成，每一個按鈕都代表了可能的使用者動作。每一個按鈕可以含有文字、圖示或兩者。您可以將工具列定義爲可浮動，讓使用者可以將工具列拖曳出畫面，進入獨立式視窗。

Property Sheet (內容表)

獨立式視窗或對話框是由標籤畫面及 OK、Cancel 及 Help 按鈕所組成。畫面資源會定義每一個標籤視窗的佈置。

Wizard (精靈)

由一系列按預定順序顯示給使用者的畫面所組的獨立式視窗或對話框，包括 Back、Next、Cancel、Finish 和 Help 按鈕。Wizard (精靈) 視窗也會在畫面左邊顯示作業列示，透過精靈來追蹤使用的進度。

Split Pane (分割窗格)

由兩個以分割列區隔的畫面所組成的次畫面。畫面可以水平或垂直排列。

Tabbed Pane (標籤窗格)

形成一標籤控制項的子窗格。此標籤控制項可以放在另一個畫面、分割窗格或重疊窗格中。

Deck Pane (重疊窗格)

由畫面集合所組成的子窗格。對於這些窗格，一次只能顯示一個畫面。例如，重疊窗格在執行時可能會變更隨著使用者的動作而顯示的畫面。

String Table (字串表格)
字串資源及其相關的資源識別字的集合。

產生檔案

畫面的可轉換字串並不儲存在 PDML 檔本身，而是儲存在另外的 Java 資源軟體組中。這些工具可讓您指定資源軟體組的定義方式，也就是 Java PROPERTIES 檔，或是 ListResourceBundle 次類別。ListResourceBundle 次類別爲可轉換資源的已編譯版，可增強 Java 應用程式的效能。不過，由於 ListResourceBundle 會於每次儲存作業中編譯，因此會導致 GUI Builder 儲存處理程序變慢。因此，最好 以 PROPERTIES 檔 (預設設定) 啓動，直到您滿意您的使用者介面設計爲止。

您可以使用工具來產生 PDML 檔中每一個畫面的 HTML 架構。在執行時間，當焦點爲在畫面的其中一個控制項上時，在使用者按一下畫面的「說明」按鈕或按下 F1 時，就會顯示正確的說明主題。您必須在 HTML 中的適當位置上插入說明內容， 也就是在 `<!-- HELPDOC:SEGMENTBEGIN -->` 及 `<!-- HELPDOC:SEGMENTEND -->` 標記範圍內。 如需其他特定說明資訊，請參閱編輯由 GUI Builder 產生的說明文件。

您可以產生將提供資料給畫面之 JavaBeans™ 的原始程式碼架構。您可以使用 GUI Builder 的「內容」視窗，爲將包含資料的控制項填寫 DATACLASS 及 ATTRIBUTE 內容。DATACLASS 內容會識別 Bean 的類別名稱，而且 ATTRIBUTE 內容會指定 Bean 類別實作之 gettor/settor 方法的名稱。一旦您新增此資訊到 PDML 檔，您便可以使用 GUI Builder 來產生 Java 原始程式碼架構並編譯它們。執行時，將呼叫適當的 gettor/settor 方法來填寫畫面的資料。

註: gettor/settor 方法的數量及類型，視該方法相關的 UI 控制項類型而定。每一個控制項的方法通訊協定都記錄在 DataBean 類別的類別說明中。

最後， 您可以將 PDML 檔案的內容序列化。 序列化可針對檔案中所有 UI 資料產生壓縮二進位表示法。 這將大大地改善使用者介面的效能，因爲 PDML 檔不必解譯， 即可顯示您的畫面。

總之： 如果已建立一個名爲 **MyPanels.pdml** 的 PDML 檔，則也會依據您在工具上選取的選項，產生下列檔案：

- **MyPanels.properties**，若您已定義資源組作爲 PROPERTIES 檔的話
- **MyPanels.java** 與 **MyPanels.class**，若您已定義資源組作爲 ListResourceBundle 次類別的話
- PDML 檔中每一畫面的 **<panel name>.html**，若您已選取要產生線上說明架構的話
- 您在 DATACLASS 內容上指定之每一個唯一 Bean 類別的 **<dataclass name>.java** 及 **<dataclass name>.class**，若您已選取要產生 JavaBeans 的原始程式碼架構
- PDML 檔中定義的每一個 UI 資源的 **<resource name>.pdml.ser**，若您選擇對它的內容進行序列化的話。

註: 如果畫面名稱和條件性行爲功能所要連接的畫面名稱相同，則條件性行爲功能 (SELECTED/DESELECTED) 將無法運作。例如，如果在 FILE1 的 PANEL1 有一個條件性行爲參照連接到某一欄位，而該欄位參照了 FILE2 中 PANEL1 的欄位，則條件性行爲事件將不會運作。若要修正此狀況，請將 FILE2 中的 PANEL1 更名，然後更新 FILE1 中的條件性行爲事件以反映此變更。

執行 Resource Script Converter

若要啓動 Resource Script Converter，請如下呼叫 Java 直譯器：

```
java com.ibm.as400.ui.tools.PDMLViewer
```

如果您未設定 CLASSPATH 環境變數來包含 Graphical Toolbox JAR 檔，則必須在指令行上，使用 `classpath` 選項來指定它們。 請參閱設置 Graphical Toolbox。

您也可以使用下列指令，在批次模式中執行 Resource Script Converter：

```
java com.ibm.as400.ui.tools.RC2XML file [options]
```

其中 *file*是要處理的資料 script (RC 檔) 的名稱。**選項**

-x name
產生的 PDML 檔案的名稱。 預設名稱爲要處理的 RC 檔的名稱。

-p name
產生的 PROPERTIES 檔案的名稱。 預設名稱爲 PDML 檔的名稱。

-r name
產生的 ListResourceBundle 子類別的名稱。 預設名稱爲 PDML 檔的名稱。

-package name
將對其指定產生之資源的資料包名稱。 若未指定，將不會產生任何資料包陳述式。

-l locale
產生資源所在的語言環境。 如果指定了語言環境，則適當的 2 字元 ISO 語言及國家或地區碼就會附加在產生的資源軟體組名稱的字尾上。

-h 產生線上說明的 HTML 架構。

-d 產生 JavaBeans 的原始程式碼架構。

-s 將所有資源序列化。

對映 Windows 資源到 PDML

所有在 RC 檔中找到的對話框、功能表及字串表，都會在已產生的 PDML 檔中轉換爲相對應的 Graphical Toolbox 資源。您也可以定義 Windows 控制項的 DATACLASS 及 ATTRIBUTE 內容，以便在您建立 Windows 資源的 ID 時，讓這些內容遵照簡單的命名慣例，傳送到新的 PDML 檔。當您執行轉換時，這些內容可以用來產生 JavaBeans 的原始程式碼架構。

Windows 資源 ID 的命名慣例爲：

```
IDCB_<class name>_<attribute>
```

其中 `<class name>` 是您想要指定爲控制項的 DATACLASS 內容的 bean 類別 完整名稱，而 `<attribute>` 則是您想要指定爲控制項的 ATTRIBUTE 內容的 bean 內容名稱。

例如，具有資源 ID `IDCB_com_MyCompany_MyPackage_MyBean_SampleAttribute` 的 Windows 文字欄位將產生 **com.MyCompany.MyPackage.MyBean** 的 DATACLASS 內容以及 **SampleAttribute** 的 ATTRIBUTE 內容。當您執行轉換時，如果您選取產生 JavaBeans，則會產生 Java 來源檔 **MyBean.java**，它含有 package 陳述式 **package com.MyCompany.MyPackage**，以及 **SampleAttribute** 內容的 gettor 與 settor 方法。

於執行時間顯示您的畫面

Graphical Toolbox 提供了可重新分送的 API，您的 Java 程式可用它來以顯示使用 PDML 定義的使用者介面畫面。API 會解譯 PDML，並使用「Java 基礎類別」來呈現您的使用者介面，藉以顯示您的畫面。

Graphical Toolbox 執行時間環境提供下列服務：

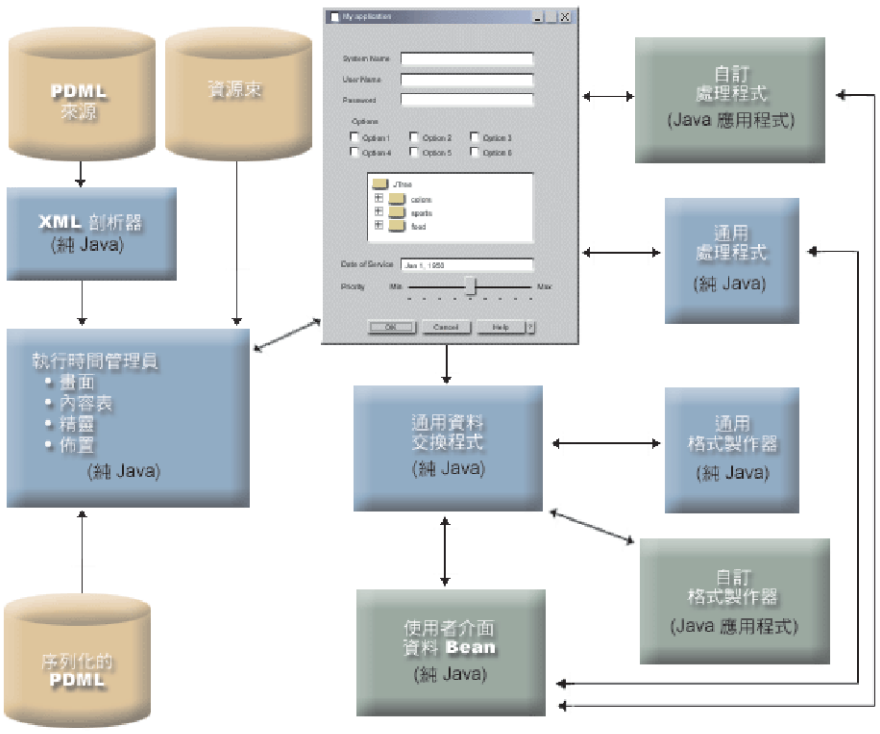
- 處理使用者介面控制項與您在 PDML 中所識別的 JavaBeans 之間的所有資料交換。
- 執行一般整數及字元資料類型的使用者資料的驗證，並定義一個介面，讓您實施自訂驗證。如果發現資料無效，將顯示錯誤訊息給使用者。
- 定義 Commit、Cancel 與 Help 事件的標準化處理，並提供一個組織架構來處理自訂事件。

- 依據 PDML 中定義的狀態資訊來管理使用者介面控制之間的互動。(例如，每當使用者選取一個特殊圓鈕時，您可能想要停用一群控制。)

com.ibm.as400.ui.framework.java 套件中含有 Graphical Toolbox 執行時間 API。

Graphical Toolbox 執行環境的元素如圖 1 所示。您的 Java 程式是**執行時間管理程式**框中一或多個物件的用戶端。

圖 1：Graphical Toolbox 執行時間環境



範例

假設畫面 **MyPanel** 定義於檔案 **TestPanels.pdml** 中，而內容檔 **TestPanels.properties** 與畫面定義相關。這兩個檔案常駐在目錄 **com/ourCompany/ourPackage** 中，可從 classpath 中定義的目錄或從 classpath 中定義的 ZIP

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

範例：建立及顯示畫面

下列程式將建立並且顯示出畫面：

```
import com.ibm.as400.ui.framework.java.*;

// Create the panel manager. Parameters:
// 1. Resource name of the panel definition
// 2. Name of panel
// 3. List of DataBeans omitted

PanelManager pm = null;
      try {
  pm = new PanelManager("com.ourCompany.ourPackage.TestPanels", "MyPanel", null);
```

```
}

catch (DisplayManagerException e) {
  e.displayUserMessage(null);
  System.exit(-1);
}

// Display the panel
pm.setVisible(true);
```

範例：建立對話框

一旦實施了提供資料給畫面的 DataBeans，以及已在 PDML 中識別了這些屬性後，下列程式可用來建構完整功能的對話框：

```
import com.ibm.as400.ui.framework.java.*;
import java.awt.Frame;

// Instantiate the objects which supply data to the panel
TestDataBean1 db1 = new TestDataBean1();
TestDataBean2 db2 = new TestDataBean2();

// Initialize the objects
db1.load();
db2.load();

// Set up to pass the objects to the UI framework
DataBean[] dataBeans = { db1, db2 };

// Create the panel manager. Parameters:
// 1. Resource name of the panel definition
// 2. Name of panel
// 3. List of DataBeans
// 4. Owner frame window

Frame owner;
...
PanelManager pm = null;
      try {
  pm = new PanelManager("com.ourCompany.ourPackage.TestPanels", "MyPanel", dataBeans, owner);
}

catch (DisplayManagerException e) {
  e.displayUserMessage(null);
  System.exit(-1);
}

// Display the panel
pm.setVisible(true);
```

範例：使用動態畫面管理程式

現存的畫面管理程式已新增一個新的服務。動態的畫面管理程式會在執行時動態調整畫面。讓我們利用動態畫面管理程式，再看一次 **MyPanel** 範例：

```
import com.ibm.as400.ui.framework.java.*;

// Create the dynamic panel manager. Parameters:
// 1. Resource name of the panel definition
// 2. Name of panel
// 3. List of DataBeans omitted

DynamicPanelManager dpm = null;
try {
  pm = new DynamicPanelManager("com.ourCompany.ourPackage.TestPanels", "MyPanel", null);
}
```

```
catch (DisplayManagerException e) {
  e.displayUserMessage(null);
  System.exit(-1);
}

// Display the panel
pm.setVisible(true);
```

當您將此畫面應用程式個體化後，您可以看到畫面的動態調整特性。請將您的游標移到 GUI 顯示畫面的邊緣，當您看見調整大小的箭頭時，您就可以變更畫面的大小。

圖 1 的長說明：Graphical Toolbox 執行時間環境 (rzahh504.gif)

位於 IBM Toolbox for Java：於執行時間顯示畫面

本圖說明 Graphical Toolbox 執行時間環境的元素如何與應用程式碼互動。

說明

本圖的組成份爲數個不同形狀、大小和色彩的箱子，以一端或兩端結尾爲箭頭的線條彼此相連。

爲了要將圖示視覺化，可以劃分成三欄以及四列，再把區域從左上往右下依序編號。例如，第一列包含區域 1、2、3；第二列包含區域 4、5、6 等等：

- 佔用區域 2、5 的對話框影像代表您 Java 程式的 GUI 介面。這個對話框的特色有各種選項，像是勾選框、文字欄位等等。
- 區域 1 頂端的兩個皮革色圓柱標示爲「PDML 來源」和「資源束」。這些圓柱代表位於儲存媒體中的 PDML 來源及 Java 資源檔。
- 區域 10 有一個皮革色圓柱標示爲 PDML Serialized，代表儲存媒體中常駐的一個或更多的序列化 PDML 檔案。
- 在對話框底端部分環繞的五個藍色矩形代表 Graphical Toolbox 的元件。從最左邊的矩形開始往逆時鐘方向移動，標示爲：
  - 區域 4 是「XML 剖析器」(Pure Java)，代表「IBM XML 剖析器」。
  - 區域 7 是「執行時間管理程式」(Pure Java)。您的 Java 程式就是「執行時間管理程式」中的一或多個物件的用戶端：「畫面」、「內容表」、「精靈」及「佈置」。
  - 區域 8 是「通用資料交換程式」(Pure Java)。
  - 區域 9 是「通用格式製作器」(Pure Java)。
  - 區域 6 是「通用處理程式」(Pure Java)。
- 三個綠色矩形代表應用程式設計師所提供的程式碼，標示如下：
  - 區域 3 是「自訂處理程式」(Java 應用程式)
  - 區域 12 是「自訂格式製作器」(Java 應用程式)
  - 區域 11 是「使用者介面資料 Bean」(Pure Java)
- 連接許多形狀的線條：
  - 單一箭頭 (一端) 的線條指示出動作。單一箭頭線條所指向的是使用線條起點處的物件的函數或元件。在以下說明中，「使用」這個詞表示一個含有單一箭頭的線條，從對物件作用的元件，指向物件。
  - 雙重箭頭 (兩端各一) 的線條指示出互動。這類線條連接著共用雙向資訊交換的物件。在以下說明中，「互動」這個字詞表示一個以雙重箭頭的線條來相互連接的元件。

您的 Java 程式的 GUI 介面 (區域 2 及 5 中的對話框影像) 與 Graphical Toolbox 的「執行時間管理程式」互動 (區域 7 中的藍色矩形)。

「執行時間管理程式」是 Pure Java，包含有畫面、內容表、精靈及 GUI 佈置。爲了產生 GUI，「執行時間管理程式」使用 Java 資源軟體組 (區域 1 中兩個皮革色圓柱的其中之一) 及 PDML 資料。「執行時間管理程式」處理 PDML 資料的方式分爲兩種：

- 使用序列化的 PDML 檔案 (區域 10 中的皮革色圓柱)
- 使用「IBM iSeries XML 剖析器」(區域 4 中的藍色矩形)，此方式會再使用 (剖析) PDML 來源檔 (區域 1 中的兩個皮革色圓柱的其中之一)

具有 GUI 功能的 Java 程式會以下列其中一種方式來操作資料：

- 讓 GUI 介面與自訂處理程式 (區域 3 中的綠色矩形) 和通用處理程式 (區域 6 中的藍色矩形) 互動
- 讓通用資料交換程式 (區域 8 中的藍色矩形) 使用 GUI 介面來取得資訊

自訂處理程式、通用處理程式和通用資料交換程式全部都和使用者介面資料 bean (區 域 11 中的綠色矩形) 互動，來回傳遞資訊。通用資料交換程式與通用格式製作器 (區 域 9 中的藍色矩形) 和自訂格式製作器 (區域 12 中的綠色矩形) 互動，將資料轉換成對於使用者介面資料 bean 來說適當的格式。

編輯由 GUI Builder 產生的說明文件

對每一個 PDML 專案檔，GUI Builder 會產生一個說明架構，並將它放入單一的 HTML 文件中。 在使用之前，針對 PDML 專案的每一個對話，此 HTML 檔會分成單一主題 的 HTML 檔。這可在使用者使用每一個主題時提供粗略的協助，並可讓您只管理 少數大型的說明檔。

Help Document 是有效的 HTML 檔，且可以在任何瀏覽器中檢視及使用大部分 的 HTML 編輯器加以編輯。Help Document 中定義區段的標示是內嵌在註解間，所以不會在瀏覽器中顯示。註解標示是用來將 Help Document 分成數段：

- 標頭
- 每一對話的主題區段
- 每一個解說啓用控制項的主題區段
- 標底

此外，您可以在標底之前新增其它的主題區段，以提供其餘資訊或一般資訊。當標頭及標底建立時，主題區段在分割之前僅有 html 主體。 當 Help Document 分割後，處理器會新增標頭及標底到主題區段，以製作完整的 HTML 檔。在 Help Document 中的標頭及標底會用來作爲預設的標頭及標底。然而，您可以以您自設的標頭置換預設的標頭。

在 Help Document 內

下列各節會說明 Help Document 的組成部分：

標頭 標頭區段的末端會顯示下列標示：

```
<!-- HELPDOC:HEADEREND -->
```

如果您想要在主題分割之後，置換所有個別主題的預設標頭，請使用 HEADER 關鍵字，並提供要併入的 html 片段的名稱。例如：

```
<!-- HELPDOC:HEADEREND HEADER="defaultheader.html" -->
```

主題區段

每一個主題都由下列標示包圍：

```
<!-- HELPDOC:SEGMENTBEGIN -->
```

和

```
<!-- HELPDOC:SEGMENTEND -->
```

緊接在 SEGMENTBEGIN 標示後的是命名區段的錨點標示。它也提供了在 Help Document 分割時所建立的 HTML 文件的檔名。區段名稱合併了畫面識別字、控制 ID 及未來的副檔名 (html)。例如：畫面的 ″MY_PANEL.MY_CONTROL.html″ 區段只有畫面識別字及未來的副檔名。

說明產生器會將文字放置於文件中，指出您放置說明資訊的位置：

```
<!-- HELPDOC:SEGMENTBEGIN PDMLSYNCH="YES" --><A NAME="MY_PANEL.MY_CONTROL.html"></A>
<H2>My favorite control</H2>
Insert help for "My favorite control" here.
<P><!-- HELPDOC:SEGMENTEND -->
```

必要時，您可以在錨點標示之後及 SEGMENTEND 標示之前新增其它

PDMLSYNCH 標籤可控制區段受制於定義在 PDML 中的控制項之程度。在 PDMLSYCH 爲 ″YES″ 的情況下， 若在 PDML 中移除與區段同名的控制項，則會同時移除 Help Document 區段。PDMLSYNCH=″NO″ 則表示，不管 PDML 中是否有對應的控制項， 主題必定會保留在 Help Document 中。例如，當您建立其他深度主題或一般主題時，就會使用此標籤。

針對畫面產生的說明鏈接至針對畫面上的說明所啓用的每個控制項。這些鏈結連同本端錨點參照一起產生，使您可以在標準瀏覽器中將它們當作內部鏈結來測試。分割 Help Document 後，處理器會移除這些內部鏈結中的 ″#″，使它們成爲結果單一主題 HTML 檔中的外部鏈結。因爲您可能需要主題內有內部鏈結，所以當參照有內含的 ″.html″ 時，處理器只會移除前置 ″#″。

如果要置換特定主題的預設標頭，請使用 HEADER 關鍵字並提供要併入的 html 片段名稱。例如：

```
<!-- HELPDOC:SEGMENTBEGIN PDMLSYNCH="YES" HEADER="specialheader.html" -->
```

標底 Help Document 中的標底會以下列標示起首：

```
<!-- HELPDOC:FOOTERBEGIN -->
```

標準標底是 </BODY></HTML>此標底會新增到每個 HTML 檔。

新增鏈接

您可以新增鏈接到任何外部或內部 URL 以及任何其它的區段。然而，您必須遵守一些慣例：

- 以標準方式使用外部 URL。這包括內部鏈接到外部 URL
- 會以標準方式撰寫同一個主題內的內部鏈結，但不可以將 ″.html″ 當作標籤名稱的一部分。這是因爲在分隔主題時，Help Document 處理器會假設任何具有 .html 的鏈接都必須是外部鏈接。 因此，它會移除前置 ″#″。
- 撰寫其他主題區段的鏈結時必須加上前置 ″#″，就好像它們是內部錨點參照。
- 也可以建立對其它主題區段的內部鏈接。只要在處理程序期間移除前面的 ″#″。

註:

- 執行時，PanelManager 類別會在具有與 PDML 檔的名稱相同的子目錄中尋找說明檔。 當處理器分割 Help Document 時，就預設值而言，它會建立此子目錄並在其中放置結果 HTML 檔。
- 處理器不會對相對鏈接的外部 URL 參照進行任何調整。當您從個別的主題檔進行鏈接時，會從新的子目錄搜尋任何相對鏈接。因此，您必須在可以找到資源的地方放置其複本，如影像，或在路徑中使用 ″../″，以便能從畫面目錄進行搜尋。

使用視效編輯程式進行編輯

您可以在幾乎任何視覺化 HTML 編輯器中編輯說明內容。 因爲 HELPDOC 標籤是註解，所以在部分編輯器中它們可能不明顯。爲方便起見，水平尺規會新增到說明架構，位置是在 SEGMENTBEGIN 標籤前面和 SEGMENTEND 標籤後面。這些水平尺規在視覺編輯器中提供整個區段的清楚視覺指示。如果因爲要移動、複製或刪除某區段而選取此區段，請選取一些周圍水平尺規以確保您的選擇會包括 SEGMENTBEGIN 和 SEGMENTEND 標籤。這些水平尺規不會複製到最終個別的 HTML 檔。

建立其它的主題

您可以在 Help Document 中建立其他主題區段。複製另一個區段通常是最容易建立其他主題區段的方法。複製區段時，您必須複製位於 SEGMENTBEGIN 標籤前面與 SEGMENTEND 標籤後面的水平尺規。這樣會使將來的視覺化編輯容易得多並可幫助避免有不符的標籤。爲取得最好的結果，請使用下列秘訣：

- 在分割 Help Document 時，錨點名稱必須是您要命名結果單一檔案的名稱。它必須是以 ″.html″ 爲結尾。
- 請在 SEGMENTBEGIN 標籤中使用 PDMLSYNCH=″NO″ 關鍵字，以避免在說明架構重新產生時移除區段。
- 新主題的任何參照會被成爲 Help Document 中的有前置 ″#″ 的內部鏈結。當區段分割成幾個單一檔案之後，在稍後的處理程序中會移除這個 ″#″。

檢查您的鏈接

就大部分撰寫而言，您可以在 Web 瀏覽器中檢視文件並選取不同鏈結來檢查鏈結。在單一 Help Document 中，鏈結仍然在它們的內部套表中。

在即將完成時，或要測試您當作對象來開發說明的應用程式時，您需要將 Help Document 分割成幾個單一檔案。您使用 Help Document to HTML Processing 來執行這項分割工作。

如果您在編輯之後需要重新產生 Help Document，您的撰寫就必須保留。如果您在產生原始解說架構後新增新的控制項，您也許想要重新產生 Help Document。在此情況下，解說產生器會在建立新的架構之前，檢查現存的 Help Document。如果有發現，它會保留任何現存的區段，然後新增新的控制項。

在瀏覽器中使用 Graphical Toolbox

您可以使用 Graphical Toolbox 爲在 Web 瀏覽器中執行的 Java Applet 建置畫面。

本節將描述如何將 Graphical Toolbox 範例中的簡單畫面轉換爲可在瀏覽器中執行的畫面。支援的瀏覽器最低等級是 Netscape 4.05 和 Internet Explorer 4.0。爲避免忙於應付個別瀏覽器的不同特性，建議您最好使用 Sun 的「Java 外掛程式」來執行 Applet。不然，您建需要建構有符號的 JAR 檔，供 Netscape 使用， 而另外建構有符號的 CAB 檔，供 Internet Explorer 使用。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

建構 applet

在 Applet 中顯示畫面的程式碼，幾乎與在 Java 應用程式範例中所使用的程式碼相同，但首先您必須將程式碼重新包在 **JApplet** 次類別的 **init** 方法中。此外，還加入了一些程式碼， 以確保 Applet 畫面的大小符合畫面的 PDML 定義所指定的維度。以下是範例 Applet **SampleApplet.java** 的原始程式碼。

```
import com.ibm.as400.ui.framework.java.*;

import javax.swing.*;
import java.awt.*;
import java.applet.*;
import java.util.*;

public class SampleApplet extends JApplet
{
    // The following are needed to maintain the panel's size
    private PanelManager        m_pm;
    private Dimension           m_panelSize;

    // Define an exception to throw in case something goes wrong
    class SampleAppletException extends RuntimeException {}

    public void init()
    {
        System.out.println("In init!");

        // Trace applet parameters
        System.out.println("SampleApplet code base=" + getCodeBase());
        System.out.println("SampleApplet document base=" + getDocumentBase());

        // Do a check to make sure we're running a Java virtual machine that's compatible with Swing 1.1
        if (System.getProperty("java.version").compareTo("1.1.5") < 0)
            throw new IllegalStateException("SampleApplet cannot run on Java VM version " +
                                            System.getProperty("java.version") +
                                            " - requires 1.1.5 or higher");

        // Instantiate the bean object that supplies data to the panel
        SampleBean bean = new SampleBean();
```

```
        // Initialize the object
        bean.load();

        // Set up to pass the bean to the panel manager
        DataBean[] beans = { bean };

        // Update the status bar
        showStatus("Loading the panel definition...");

        // Create the panel manager. Parameters:
        // 1. PDML file as a resource name
        // 2. Name of panel to display
        // 3. List of data objects that supply panel data
        // 4. The content pane of the applet

        try { m_pm = new PanelManager("MyGUI", "PANEL_1", beans, getContentPane()); }
        catch (DisplayManagerException e)
        {
            // Something didn't work, so display a message and exit
            e.displayUserMessage(null);
            throw new SampleAppletException();
        }

        // Identify the directory where the online help resides
        m_pm.setHelpPath("http://MyDomain/MyDirectory/");

        // Display the panel
        m_pm.setVisible(true);
    }

    public void start()
    {
        System.out.println("In start!");

        // Size the panel to its predefined size
        m_panelSize = m_pm.getPreferredSize();
        if (m_panelSize != null)
        {
            System.out.println("Resizing to " + m_panelSize);
            resize(m_panelSize);
        }
        else
            System.err.println("Error: getPreferredSize returned null");
    }

    public void stop()
    {
        System.out.println("In stop!");
    }

    public void destroy()
    {
        System.out.println("In destroy!");
    }

    public void paint(Graphics g)
    {
        // Call the parent first
        super.paint(g);

        // Preserve the panel's predefined size on a repaint
        if (m_panelSize != null)
            resize(m_panelSize);
    }
}
```

applet 的內容畫面將傳給 Graphical Toolbox，作爲要佈置的儲存區。在 **start** 方法中，將 applet 窗格調整至正確的大小，然後置換 **paint** 方法，以便在重新調整瀏覽器視窗的大小時，能夠保留畫面的大小。

於瀏覽器中執行 Graphical Toolbox 時，無法自 JAR 檔中存取畫面的線上 說明 HTML 檔。它們必須以個別的檔案常駐在目錄中，而該目錄則有您的 Applet 常駐。對 **PanelManager.setHelpPath** 所進行的呼叫會把這個目錄識別成「圖形化工具箱」，於是就能找到您的說明檔。

HTML 標記

由於我們建議您使用 Sun 的「Java 外掛程式」來提供 Java 執行時間環境的正確層次，因此識別 Graphical Toolbox Applet 的 HTML 不如預期般的直接明確。幸運的是，僅需做小小的更動，即可以對另一個 Applet 重複使用同一個 HTML 模版。根據設計，標記在 Netscape Navigator 及 Internet Explorer 中都可解譯，而且如果使用者的電腦中未安裝「Java 外掛程式」，它會產生提示，告知使用者由 Sun 的網站中下載它。如需「Java 外掛程式」作業的詳細資訊，請參閱 Java 外掛程式 HTML 規格。 

以下是範例 applet 的 HTML，位於檔案 **MyGUI.html** 中：

```
<html>

<head>
<title>Graphical Toolbox Demo</title>
</head>

<body>
<h1>Graphical Toolbox Demo Using Java(TM) Plug-in</h1>
<p>

<!-- BEGIN JAVA(TM) PLUG-IN APPLET TAGS -->

<!-- The following tags use a special syntax which allows both Netscape and Internet Explorer to load -->
<!-- the Java Plug-in and run the applet in the Plug-in's JRE.  Do not modify this syntax.            -->
<!-- For more information see http://java.sun.com/products/jfc/tsc/swingdoc-current/java_plug_in.html.-->

<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
        width="400"
        height="200"
        align="left"
        codebase="http://java.sun.com/products/plugin/1.1.3/jinstall-113-win32.cab#Version=1,1,3,0">
    <PARAM name="code"     value="SampleApplet">
    <PARAM name="codebase" value="http://www.mycompany.com/~auser/applets/">
    <PARAM name="archive"  value="MyGUI.jar,jui400.jar,util400.jar,x4j400.jar">
    <PARAM name="type"     value="application/x-java-applet;version=1.1">

    <COMMENT>
    <EMBED type="application/x-java-applet;version=1.1"
           width="400"
           height=200"
           align="left"
           code="SampleApplet"
           codebase="http://www.mycompany.com/~auser/applets/"
           archive="MyGUI.jar,jui400.jar,util400.jar,x4j400.jar"
           pluginspage="http://java.sun.com/products/plugin/1.1.3/plugin-install.html">
        <NOEMBED>
    </COMMENT>
                No support for JDK 1.1 applets found!
        </NOEMBED>
    </EMBED>
</OBJECT>

<!-- END JAVA(TM) PLUG-IN APPLET TAGS -->
```

```
<p>
</body>
</html>
```

設定 1.1.3 的版本資訊是很重要的。

註: 此範例中，XML 剖析器 JAR 檔案 **x4j400.jar** 儲存在 Web 伺服器上。您可以改用其他 XML 剖析器。如需詳細資訊，請參閱第 370 頁的『XML 剖析器與 XSLT 處理器』。唯有 PDML 檔案併入 applet 安裝的一部分時才有必要這樣儲存。因爲效能上的理由，通常應該*序列化*畫面定義，以便 Graphical Toolbox 不必在執行時間解譯 PDML。如此一來，可建立畫面的二進位表示法，大幅增進使用者介面的效能。如需更多資訊，請參閱工具所產生的檔案的說明。

安裝及執行 applet

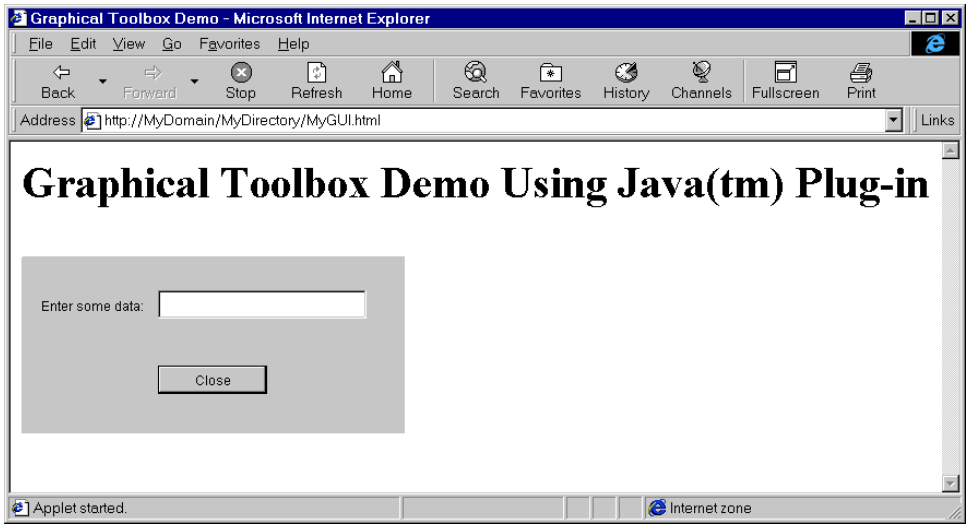
您可以執行下列步驟，在您喜愛的 Web 伺服器上安裝 applet：

1. 編譯 **SampleApplet.java**。
2. 建立一個名爲 **MyGUI.jar** 的 JAR 檔，來包含 applet 二進位。 包括當您編譯 **SampleApplet.java** 和 **SampleBean.java**、PDML 檔 **MyGUI.pdml** 和資源組 **MyGUI.properties** 時所產生的類別檔。
3. 將新的 JAR 檔複製到您在 Web 伺服器上選擇的目錄。將含有線上說明的 HTML 檔複製到伺服器目錄中。
4. 將 Graphical Toolbox JAR 檔複製到伺服器目錄中。
5. 最後，將內含 applet 的 HTML 檔 **MyGUI.html** 複製到伺服器目錄中。

提示: 測試 Applet 時，請確定您已從工作站的 CLASSPATH 環境變數中，移除了 Graphical Toolbox jar。否則您會看到錯誤訊息，表示在伺服器中找不到 Applet 的資源。

現在您可以執行 applet。請將 Web 瀏覽器指向伺服器上的 **MyGUI.html**。如果您尚未安裝「Java 外掛程式」，系統將詢問您是否要加以安裝。一旦安裝了外掛程式且啓動了 applet 後， 您的瀏覽器畫面將類似圖 1：

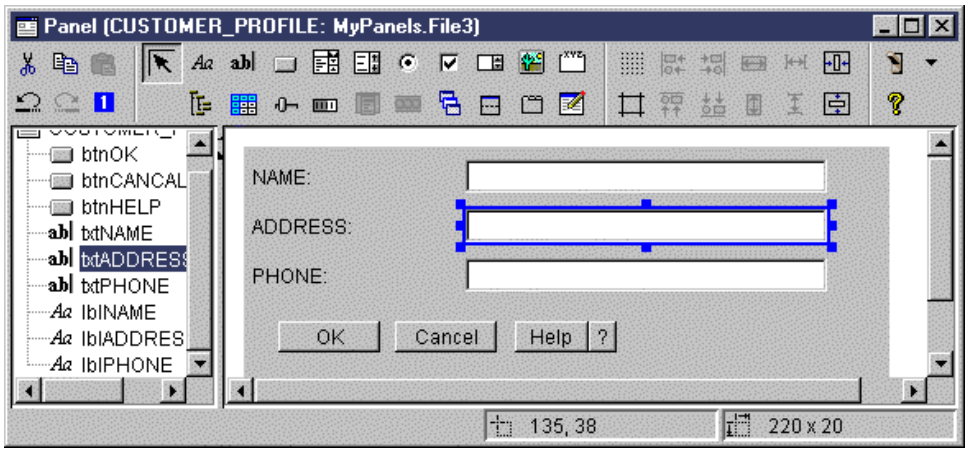
圖 1：在瀏覽器中執行範例 applet



GUI Builder 畫面建置器工具列

圖 1 顯示「GUI 建置器畫面建置器」視窗。在圖 1 之後則是一份清單，它列出每一個「畫面建置器」工具圖示及其功能的說明。

圖 1：「GUI 建置器畫面」視窗



 按一下「指標」可以移動及調整畫面元件的大小。

 按一下「標籤」可以在畫面上插入靜態標籤。

 按一下「本文」可以在畫面上插入文字框。

 按一下「按鈕」可以在畫面上插入按鈕。

 按一下「組合框」可以在畫面上插入下拉清單框。

 按一下「清單框」可以在畫面上插入清單框。

 按一下「圓鈕」可以在畫面上插入圓鈕。

 按一下「勾選框」可以在畫面上插入勾選框。

 按一下「調整器」可以在畫面上插入調整器。

 按一下「影像」可以在畫面上插入影像。

 按一下「功能表列」可以在畫面上插入功能表列。

 按一下「群組框」可以在畫面上插入含標示的群組框。

 按一下「樹」可以在畫面上插入階層樹。

 按一下「表格」可以在畫面上插入表格。

 按一下「滑動區」可以在畫面上插入可調整的滑動區。

 按一下「進度列」可以在畫面上插入進度列。

 按一下「疊窗格」可以在畫面上插入疊窗格。一個疊窗格包含一疊畫面。使用者可以選取其中的任何畫面，但只能完全看到選取的畫面。

 按一下「分割窗格」可以在畫面上插入分割窗格。一個分割窗格可分成兩個水平窗格或垂直窗格。

 按一下「標籤窗格」可以在畫面上插入含標籤的窗格。一個含標籤窗格含有一組畫面，每個畫面的頂端有標籤。使用者可以按一下標籤來顯示畫面的內容。會以畫面的標題作爲標籤的文字。

 按一下「自訂」可以在畫面上插入自行定義的使用者介面元件。

 按一下「工具列」可以在畫面上插入工具列。

 按一下「輪換格線」可以在畫面上啓用格線。

 按一下「向上對齊」，可以將畫面上的多個元件對齊特定或主要元件的頂端。

 按一下「向下對齊」，可以將畫面上的多個元件對齊特定或主要元件的底邊。

 按一下「等高」，可以將多個元件的高度對齊特定或主要元件的高度。

 按一下「垂直置中」，可以將選取的元件垂直地放在畫面的中央。

 按一下「輪換邊距」可以檢視畫面的邊距。

 按一下「靠左對齊」，可以將畫面上的多個元件對齊特定或主要元件的左邊。

 按一下「靠右對齊」，可以將畫面上的多個元件對齊特定或主要元件的右邊。

 按一下「等寬」，可以將多個元件的寬度對齊特定或主要元件的寬度。

 按一下「水平置中」，可以將選取的元件水平地放在畫面的中央。

 按一下「剪下」可以剪下畫面元件。

 按一下「複製」按鈕可以複製畫面元件。

 按一下「貼上」可以在不同的畫面或檔案之間貼上畫面元件。

 按一下「還原」可以還原上一個動作。

 按一下「重做」可以重做上一個動作。

 按一下「標籤次序」，即可在使用者按下 TAB 鍵來導覽畫面時，控制每一個元件的選擇次序。

 按一下「預覽」可以顯示畫面的外觀預覽。

 按一下「說明」可以取得有關 Graphical Toolbox 的更多特定資訊。

IBM Toolbox for Java Bean

JavaBeans 是以 Java 編寫並可重複使用的軟體元件。元件是程式的一部分， 它會提供定義良好的功能單元，這種單元可小至視窗中按鈕的標籤，或大至整個應用程式。

JavaBeans 可以是視覺化或非視覺化的元件。非視覺化 JavaBeans 仍具有視覺化的呈現方式 (例如圖示或名稱)，以允許視覺化操作。

許多 IBM Toolbox for Java 公用類別也是 JavaBeans。這些類別是依據 Javasoft JavaBean 標準來建立的；因此它們可作爲重複使用的元件。IBM Toolbox for Java Bean 的內容及方法與類別的內容及方法相同。

JavaBeans 可以在應用程式內使用，或可在建置器工具 (例如 IBM VisualAge for Java 產品) 中以視覺化方式操作它們。

範例

下列範例說明如何在程式中使用 JavaBeans，以及如何使用視覺化 Bean 建置器從 JavaBeans 建立程式：

第 486 頁的『範例：IBM Toolbox for Java Bean 程式碼』

第 487 頁的『範例：使用視覺化 bean 建置器來建立 bean』

JDBC

JDBC 是 Java 平台所附的應用程式設計介面 (API)，可讓 Java 程式連接多種資料庫。

IBM Toolbox for Java JDBC 驅動程式可讓您使用 JDBC API 介面，以對伺服器上的資料庫發出結構化查詢語言 (SQL) 陳述式，並處理來自伺服器上資料庫的結果。您也可以使用 IBM Developer Kit for Java JDBC 驅動程式，稱爲「原有的」JDBC 驅動程式：

- 當 Java 程式在某個系統上而資料庫檔案是在另一個系統上時 (如在主從架構環境中)，請使用 IBM Toolbox JDBC 驅動程式。
- 當 Java 程式及資料庫檔案在同一個 iSeries 伺服器上時，請使用原有的 JDBC 驅動程式

如需 IBM Toolbox for Java JDBC 類別及範例、目前進行的改良、JDBC 內容及未支援之 SQL 類型等的相關資訊，請參閱下列頁面：

第 59 頁的『JDBC 類別』

第 292 頁的『版本 5 版次 3 的 JDBC 支援加強功能』

第 295 頁的『IBM Toolbox for Java JDBC 內容』

第 308 頁的『JDBC SQL 類型』

JDBC 的不同版本

JDBC API 有不同的版本，IBM Toolbox for Java JDBC 驅動程式支援下列版本：

- JDBC 1.2 API (java.sql 套件) 包含在 Java Platform 1.1 核心 API 及 JDK 1.1 中。
- JDBC 2.1 核心 API (java.sql 套件) 包含在 Java 2 Platform 標準版 (J2SE) 及 Java 2 Platform 企業版 (J2EE) 中。
- JDBC 2.0 Optional Package API (javax.sql 套件) 包括於 J2EE 中且當作 Sun 的個別下載來使用。這些延伸程式以前稱爲 JDBC 2.0 Standard Extension API。
- JDBC 3.0 API (java.sql 和 javax.sql 套件) 包括在 J2SE 1.4 版中。

V5R4 之 IBM Toolbox for Java JDBC 支援的加強功能

i5/OS 版本 5 版次 4 已加強數個 JDBC 功能。

i5/OS 版本 5 版次 4 的加強 JDBC 功能包括：

- 『2 MB 陳述式大小』
- 『128 位元組直欄名稱支援』
- 第 292 頁的『資料庫主電腦伺服器追蹤支援』
- 第 292 頁的『eWLM 相關因子支援』

如需有關前版次加強 JDBC 功能的相關資訊，請參閱第 292 頁的『版本 5 版次 3 的 JDBC 支援加強功能』及第 293 頁的『i5/OS 版本 5 版次 2 的 JDBC 加強功能』。

2 MB 陳述式大小

在 V5R4 之前，SQL 陳述式大小的限制爲 65 535 位元組。當陳述式文字使用單位元組 CCSID 表示時與 65 535 字元對應，當陳述式文字使用雙位元組 CCSID 表示時與 32 767 字元對應。一些客戶，特別是使用自動產生 SQL 陳述式之應用程式的那些客戶，會受到受此限制影響。

在 V5R4 中，iSeries 陳述式大小限制增加到 2 MB (或 2 097 152 位元組)。IBM Toolbox for Java JDBC 驅動程式總是以雙位元組 Unicode 傳送陳述式文字。因此，陳述式長度的最大值 (字元) 是 1 MB (或 1 048 576 字元)。

128 位元組直欄名稱支援

從 V5R4 開始，資料庫將支援 SQL 表格使用高達 128 位元組的直欄名稱。在 V5R4 之前，最多支援 30 位元組的直欄名稱。IBM Toolbox for Java JDBC 驅動程式會將這些可能更長的名稱提供給其使用者。

有一個例外情況是不會傳回 128 位元組直欄名稱。當使用本端資料包快取，且直欄名稱超過 30 個字元時，伺服器會將直欄名稱作爲系統直欄名稱傳回。

資料庫主電腦伺服器追蹤支援

已經將一個選項新增至 Toolbox for Java JDBC 驅動程式，以開啓資料庫主電腦伺服器追蹤。爲了支援此功能，已將選項 "64" 新增至 "伺服器追蹤" 連線內容。如需詳細資料，請參閱 第 295 頁的『IBM Toolbox for Java JDBC 內容』。

eWLM 相關因子支援

IBM Toolbox for Java 會接受 IBM Enterprise Workload Manager (eWLM) 相關因子，並將其作爲連線屬性相關因子傳遞至主電腦，以使用「應用程式回應測量 (ARM) API」。在使用 AS400JDBCConnection 類別中的下列方法建立連線之後，此相關因子可以在任何時間傳送至主電腦：

setDB2eWLMCorrelator

```
public void setDB2eWLMCorrelator(byte[] bytes)
                          throws SQLException
```

設定「eWLM 相關因子」。假設使用有效的相關因子值。如果是空值，將關閉所有 ARM/eWLM 實作。eWLM 相關因子需要 i5/OS V5R3 或更新版本伺服器。當執行至 OS/400® V5R2 或之前版本伺服器時，會忽略此要求。

參數：

- `bytes`：eWLM 相關因子值
- `SQLException`：請參閱 Sun Microsystems 公司網站的 Class SQLException 資訊。

如需前版次加強 JDBC 功能的相關資訊，請參閱『版本 5 版次 3 的 JDBC 支援加強功能』及第 293 頁的『i5/OS 版本 5 版次 2 的 JDBC 加強功能』。

版本 5 版次 3 的 JDBC 支援加強功能

i5/OS 版本 5 版次 3 提升了幾項 JDBC 功能。

i5/OS 版本 5 版次 3 的 JDBC 加強功能包括：

- UTF-8 與 UTF-16 支援
- Binary 與 Varbinary 支援
- 加入小數位數支援
- 2 GB 大型物件支援：
- Insensitive 游標支援
- 具體化的查詢表格支援

如需前版次 JDBC 加強功能的相關資訊，請參閱 IBM Toolbox for Java JDBC 支援的 V5R2 加強功能。

UTF-8 與 UTF-16 支援

UTF-8 資料儲存在字元欄位中，CCSID 爲 1208。UTF-8 字元是非合併字元的可變位元組數 (1、2、3 或 4)，也可以是合併字元的任何位元組數。指定給字元欄位的長度代表此欄位可以包含的最大位元組數。您可以使用 UTF-8 1208 CCSID 來標示下列資料類型：

- 固定長度字元 (CHAR)
- 可變長度字元 (VARCHAR)
- 字元 LOB (CLOB)

UTF-16 資料儲存在圖形欄位中，CCSID 爲 1200。就非合併字元而言，UTF-16 字元的長度可以是兩個或四個位元組 (亦即 Surrogate)，若爲合併字元，則可爲任何位元組數。指定給圖形資料欄位的長度代表此欄位可以包含的最大雙位元組字元數。您可以使用 UTF-16 1200 CCSID 來標示下列資料類型：

- 固定長度圖形 (GRAPHIC)
- 可變長度圖形 (VARGRAPHIC)
- 雙位元組字元 LOB (DBCLOB)

Binary 與 Varbinary 支援

BINARY 和 VARBINARY 資料類型類似於 CHAR 和 VARCHAR 資料類型，但包含二進位資料而非字元資料。BINARY 欄位有固定長度。VARBINARY 欄位的長度可變。BINARY 和 VARBINARY 資料類型有下列特性：

- 二進位類型的編碼字集 ID (CCSID) 是 65535
- 在分派與比較方面，二進位資料類型只與其他二進位資料類型 (BINARY、VARBINARY 和 BLOB) 相容
- 二進位資料類型的填充字元是 x'00'，而非空白字元
- 需要將尾端字元折行以防止截斷情形發生時，會折行的是 x'00' 字元，而非尾端空白。
- 比較二進位資料類型時，必須要欄位中的資料與長度完全相同，才表示兩個欄位相等。在比較時，尾端的零不會被忽略
- 在比較二進位資料類型時，若兩個欄位的長度不同，則較短的欄位被視爲小於較長的欄位。

加入小數位數支援

小數位數現在可支援多達 63 個位數。AS400JDBCDataSource 中加入了、「最大有效位數」和「最大小數位數」這三個內容，以及 setMinimumDivideScale(int divideScale)、getMinimumDivideScale()、setMaximumPrecision(int precision)、getMaximumPrecision()、setMaximumScale(int scale) 和 getMaximumScale() 六個方法。最小除法小數位數可指定十進位除法結果的最小小數位數值，可設定爲 0 到 9 之間的任何數。最大有效位數可指定資料庫使用的最大有效位數，可設定爲 31 或 63。最大有效位數可指定資料庫所使用的最大有效位數，可設定 0 到 63 之間的任何數。

2 GB 大型物件 (LOB) 支援

IBM Toolbox for Java JDBC 的加強功能目前可支援高達 2 GB LOB 的使用量。

Insensitive 游標支援

游標支援目前可支援 insensitive 游標。使用具有 TYPE_SCROLL_INSENSITIVE 的 ResultSet， 會使用 insensitive 游標。基礎資料庫開啓時，ResultSet 不會顯示其變更。

具體化的查詢表格支援

將 ″MATERIALIZED QUERY TABLE″ 傳回作爲對 DatabaseMetaData.getTables() 的呼叫時的 TABLE_TYPE。

i5/OS 版本 5 版次 2 的 JDBC 加強功能

i5/OS 版本 5 版次 2 的一些 JDBC 功能已有所加強。

i5/OS 版本 5 版次 2 的 JDBC 加強功能包括：

- FOR UPDATE 限制的移除
- 資料截斷的變更
- 依名稱取得與修改直欄及參數
- 擷取自動產生的鍵值

- 改進於批次中執行 SQL insert 陳述式的效能
- 強化對 ResultSet.getRow() 的支援
- 增進對直欄名稱使用大小寫的支援
- 指定 Statements、CallableStatements 及 PreparedStatements 的保留能力
- 強化異動隔離支援

FOR UPDATE 限制的移除

您不再需要於 SELECT 陳述式中指定 FOR UPDATE 以保證游標可更新。連接至 i5/OS V5R1 與以上的版本時，對於您在建立陳述式時所傳入的任何並行，IBM Toolbox for Java 都可處理。如果不指定並行性，則預設值仍爲唯讀游標。

資料截斷僅於截斷字元資料寫入資料庫時，才會擲出異常訊息

目前，IBM Toolbox for Java 的資料截斷規則與 IBM Developer Kit for Java JDBC 驅動程式的資料截斷規則相同。如需相關資訊，請參閱 IBM Toolbox for Java JDBC 內容。

依名稱取得與修改直欄及參數

新的方法可讓您依 ResultSet 中的直欄名稱取得與更新資訊， 及依 CallableStatement 中的參數名稱取得與設定資訊。 例如，在 ResultSet 中，先前您是使用下列項目：

```
ResultSet rs = statement.executeQuery( SELECT * FROM MYCOLLECTION/MYTABLE );
     rs.getString(1);
```

現在您可使用：

```
ResultSet rs = statement.executeQuery( SELECT * FROM MYCOLLECTION/MYTABLE );
     rs.getString( 'STUDENTS' );
```

請注意，依參數的索引存取參數之效能較依參數名稱存取參數的效能爲佳。您也可指定在 CallableStatement 中設定的參數名稱。先前在 CallableStatement 中您可能使用：

```
CallableStatement cs = connection.prepareCall( CALL MYPGM (?) );
cs.setString( 1 );
```

現在您可使用：

```
CallableStatement cs = connection.prepareCall( CALL MYPGM (?) );
cs.setString( 'PARAM_1' );
```

若要使用這些新的方法，需要 JDBC 3.0 或更新版本及 Java 2 Platform 1.4 版 (標準版或企業版)。

擷取自動產生的鍵值

AS400JDBCStatement 中陳述式物件不產生任何鍵值時，會傳回空的 ResultSet 物件。 目前伺服器僅支援傳回一個自動產生的鍵值 (最後插入的列之鍵值)。下列範例告訴您如何將值插入表格後取得自動產生的鍵值：

```
Statement s =
   statement.executeQuery("INSERT INTO MYSCHOOL/MYSTUDENTS (FIRSTNAME) VALUES ('JOHN'");
ResultSet rs = s.getGeneratedKeys();
     // Currently the iSeries server supports returning only one auto-generated
     // key -- the key for the last inserted row.
rs.next();
String autoGeneratedKey = rs.getString(1);
     // Use the auto-generated key, for example, as the primary key in another table
```

若要擷取自動產生的鍵值，需要 JDBC 3.0 或更新版本，以及 Java 2 Platform 1.4 版 (標準版或企業版)。擷取自動產生的鍵值也需與 i5/OS V5R2 或更新版本連接。

改進於批次中執行 SQL insert 陳述式的效能

於批次中執行 SQL insert 陳述式的效能已改進。於批次中執行 SQL 陳述式的方法爲： 使用下列項目

中不同的 addBatch() 方法：AS400JDBCStatement、AS400JDBCPreparedStatement 及 AS400JDBCCallableStatement。強化的批次支援僅會影響 insert 要求。例如，使用批次支援來處理數個僅牽涉對伺服器一次傳送的 insert。 不過，使用批次支援來處理一個 insert、update 及一個 delete，則會分別傳送各要求。

若要使用批次支援，需要 JDBC 2.0 或更新版本及 Java 2 Platform 1.2 版 (標準版或企業版)。

強化對 ResultSet.getRow() 的支援

之前，IBM Toolbox for Java JDBC 驅動程式受限於 ResultSet 中 getRow() 方法的支援。特別是使用具負值的 ResultSet.last()、ResultSet.afterLast() 及 ResultSet.absolute() 會使目前的列號無法使用。現在已將此限制移除，因而使此方法的功能更全面性。

對直欄名稱使用大小寫

IBM Toolbox for Java 方法必須讓使用者提供的直欄名稱，或應用程式提供的直欄名稱，與資料庫表格上的名稱相符。不論是哪一種情況，當直欄名稱未以引號括住時，IBM Toolbox for Java 都會於名稱與伺服器上的名稱比對之前，將名稱變更爲大寫字元。而當直欄名稱以引號括住時，必須完全與伺服器上的名稱相符，否則 IBM Toolbox for Java 會丟出異常。

指定建立的 Statements、CallableStatements 及 PreparedStatements 之保留能力

AS400JDBCConnection 中的新方法可讓您指定您所建立的 Statements、CallableStatements當確定異動時，保留能力決定游標是否維持在開啓或是關閉。現在您會具有與其連線物件不同保留能力的陳述式。此外，連線物件可具有多個開啓的陳述式物件，每一個具有不同的指定保留能力。呼叫 commit 會使系統根據指定給該陳述式的保留能力來處理每一個陳述式。

保留能力以下列順序衍生：

1. 使用 Connection 類別方法 createStatement()、prepareCall() 或 prepareStatement() 建立陳述式時指定的保留能力。
2. 使用 Connection.setHoldability(int) 指定的保留能力。
3. IBM Toolbox for Java JDBC 游標保留內容指定的保留能力 (未使用 1. 或 2. 中的方法時)

若要使用這些新的方法，需要 JDBC 3.0 或更新版本，以及 Java 2 Platform 1.4 版 (標準版或企業版)。此外，執行 i5/OS V5R1 或之前版本的伺服器僅可使用由 JDBC 游標保留內容指定的保留能力。

強化異動隔離支援

IBM Toolbox for Java JDBC 驅動程式現在支援連接後切換至異動隔離層次 *NONE。在 V5R2 之前，IBM Toolbox for Java JDBC 驅動程式在連接後切換至 *NONE 時會丟出異常。

IBM Toolbox for Java JDBC 內容

使用 JDBC 連接伺服器資料庫時可指定許多內容。所有的內容都是可選用的， 且可將之指定爲 URL 的一部分，或是在 java.util.Properties 物件中指定。 若同時在 URL 及 Properties 物件中設定內容，將會使用 URL 中的值。

註: 下面清單不包括 DataSource 內容。

下列表格列出由此驅動程式所辨識之不同的連線內容。這些內容中的部分內容會影響效能，其它的則是伺服器工作屬性。表格將內容組織爲下列種類：

- 第 296 頁的『一般內容』
- 第 296 頁的『伺服器內容』
- 第 299 頁的『格式內容』
- 第 300 頁的『效能內容』
- 第 302 頁的『排序內容』

- 第 303 頁的『其它內容』

一般內容

一般內容是指定使用者、密碼及連接伺服器時是否需要提示的系統屬性。

一般內容	說明	必要的	選項	預設
″密碼″	指定連接伺服器的密碼。若未指定密碼，則會提示使用者，除非 ″提示″ 內容設定爲 ″false″ (在這種情形下嘗試連線會失敗)。	否	伺服器密碼	(將會提示使用者)
″提示″	指定在連接伺服器時若需要使用者名稱或密碼，是否要提示使用者。如果未提示使用者就無法連線，而此內容又設定爲 ″false″，則連線嘗試會失敗。	否	″true″ ″false″	″true″
″使用者″	指定用於連接伺服器的使用者名稱。若未指定密碼，則會提示使用者，除非 ″提示″ 內容設定爲 ″false″ (在這種情形下嘗試連線會失敗)。	否	伺服器使用者	(將會提示使用者)

伺服器內容

伺服器內容指定支配異動、檔案庫及資料庫的屬性。

伺服器內容	說明	必要的	選項	預設
″游標保留″	指定是否在異動內保留游標。如果此內容設定爲 ″true″，則在確定或回轉異動後，游標不會關閉。在工作單元期間所擷取的所有資源都將保留，但會釋放工作單元期間所隱含擷取之特定列及物件的鎖定。	否	″true″ ″false″	″true″
″游標靈敏度″	指定資料庫要求的游標靈敏度。此行爲取決於 resultSetType： • ResultSet.TYPE_FORWARD_ONLY 或 ResultSet.TYPE_SCROLL_SENSITIVE 表示此內容值可控制 Java 程式從資料庫中所要求的游標靈敏度。 • ResultSet.TYPE_SCROLL_INSENSITIVE 則會忽略此內容。 連接到執行 i5/OS V5R1 及先前版本的系統時，會忽略此內容。	否	″″ (使用 ResultSet Type 來決定游標靈敏度) ″asensitive″ ″sensitive″ ″insensitive″	″″
″資料庫名稱″	指定連線使用的資料庫，其中包括儲存於獨立輔助儲存體儲存區 (ASP) 的資料庫。此內容僅適用於連接到 i5/OS 的 V5R2 或更新版本。指定資料庫名稱時，該名稱必須存在於伺服器上的關聯式資料庫目錄中。下列的基準判定所存取的資料庫： • 當使用此內容指定資料庫時，所指定的資料庫已被使用。指定的資料庫不存在時則連線失敗。 • 當使用此內容指定 *SYSBAS 爲資料庫名稱時，則使用系統預設資料庫。 • 當省略此內容時，則使用在使用者設定檔之工作說明中指定的資料庫名稱。當工作說明未指定資料庫名稱時，則使用系統預設資料庫。	否	資料庫名稱 ″*SYSBAS″	使用在使用者設定檔之工作說明中指定的資料庫名稱。當工作說明未指定資料庫名稱時，則使用系統預設資料庫。

伺服器內容	說明	必要的	選項	預設
″檔案庫″	指定您要在伺服器工作的檔案庫清單中新增或置換的一或多個檔案庫，並選擇性地設定預設檔案庫 (預設綱目)。 **檔案庫清單** 伺服器會使用指定的檔案庫，解析不合格的儲存程序名稱，然後儲存程序會使用它們來解析不合格名稱。若要指定多個檔案庫，請使用逗點或空格來分隔個別登錄。您可以使用 *LIBL 作爲伺服器工作之現行檔案庫清單的位置保留符號： • 第一個登錄是 *LIBL 時，指定的檔案庫會新增到伺服器工作的現行檔案庫清單 • 若未使用 *LIBL，則指定的檔案庫會置換伺服器工作的現行檔案庫清單 有關檔案庫清單內容的資訊，請參閱 JDBC LibraryList 內容。 **預設綱目** 伺服器會使用預設綱目來解析 SQL 陳述式中不合格的名稱。例如，在陳述式 ″SELECT * FROM MYTABLE″ 中，伺服器只會在預設綱目中尋找 MYTABLE。您可以指定連線 URL 中的預設綱目。 當您沒有指定連線 URL 中的預設綱目時，則會套用下列條件，視您是否使用「SQL 命名」或「系統命名」而定。 • **SQL 命名** 若未在連線 URL 中指定預設綱目： – 第一個登錄 (除非它是 *LIBL) 會變成預設綱目 – 當第一個登錄是 *LIBL 時，第二個登錄會變成預設綱目 – 當您沒有設定此內容或當它只有 *LIBL 時，使用者設定檔會變成預設綱目 • **系統命名** 若未在連線 URL 中指定預設綱目： – 沒有設定任何預設綱目，且伺服器會使用指定的檔案庫來搜尋未限定的名稱 – 當您沒有設定此內容或當它只有 *LIBL 時，伺服器會使用伺服器工作的現行檔案庫清單來搜尋未限定的名稱	否	伺服器檔案庫清單，以逗點或空格區隔	″*LIBL″
″最大有效位數″	指定資料庫可以使用的最大有效位數。	否	″31″ ″63″	″31″
″最大小數位數″	指定資料庫可以使用的最大小數位數。	否	″0″-″63″	″31″

伺服器內容	說明	必要的	選項	預設
″最小除法小數位數″	指定十進位除法結果的最小小數位數值。	否	″0″ ″1″ ″2″ ″3″ ″4″ ″5″ ″6″ ″7″ ″8″ ″9″	″0″
″資料包 ccsid″	指定要用於 SQL 資料包的字元編碼以及要傳送到伺服器的任何陳述式。	否	″1200″ (UCS-2) ″13488″ (UTF-16)	″13488″
″回轉保留游標″	指定是否在回轉後保留游標。如果此內容設爲 ″true″，則在回轉異動後會保留游標。	否	″true″ ″false″	″false″
″異動隔離″	指定預設異動隔離。	否	″無″ ″讀取未確定″ ″讀取已確定″ ″可重複讀取″ ″可序列化的″	″讀取未確定″
″轉換十六進位″	指定如何解譯十六進位文字。	否	″字元″ (將十六進位字元文字解譯成字元資料) ″二進位″ (將十六進位字元文字解譯成二進位資料)	″字元″
″true 自動確定″	指定連線是否應使用 true 自動確定支援。true 自動確定表示，自動確定已開啓且正在 *NONE 之外的隔離層次下執行。依預設，驅動程式會藉由在 *NONE 伺服器隔離層次下執行，以處理自動確定。	否	″true″ (使用 true 自動確定。) ″false″ (不使用 true 自動確定。)	″false″

伺服器內容	說明	必要的	選項	預設
″xa 鬆散耦合支援″	指定鬆散耦合異動分支是否允許鎖定共用。 **註:** 當在 i5/OS V5R3 或之前的版本上執行時，會忽略此設定。	否	″0″ = 無法共用鎖定 ″1″ = 可以共用鎖定	″0″

格式內容

格式內容指定日期及時間格式、日期及小數點符號，以及 SQL 陳述式內用到的表格命名慣例。

格式內容	說明	必要的	選項	預設
″日期格式″	指定 SQL 陳述式內日期文字使用的日期格式。	否	″mdy″ ″dmy″ ″ymd″ ″usa″ ″iso″ ″eur″ ″jis″ ″julian″	(伺服器工作)
″日期分隔字元″	指定 SQL 陳述式內日期文字使用的日期分隔字元。除非 ″日期格式″ 內容設定爲 ″julian″、″mdy″、″dmy″ 或 ″ymd″，否則此內容無效。	否	″/″ (斜線) ″-″ (破折號) ″.″ (句點) ″,″ (逗點) ″b″ (空格)	(伺服器工作)
″小數分隔字元″	指定 SQL 陳述式內數字文字使用的小數點符號。	否	″.″ (句點) ″,″ (逗點)	(伺服器工作)
″命名″	指定參照表格時使用的命名慣例。	否	″sql″ (像是在 `schema.table` 中) ″系統″ (像是在 `schema/table` 中)	″sql″
″時間格式″	指定 SQL 陳述式內時間文字使用的時間格式。	否	″hms″ ″usa″ ″iso″ ″eur″ ″jis″	(伺服器工作)
″時間分隔字元″	指定 SQL 陳述式內時間文字使用的時間分隔符號。 除非 ″時間格式″ 內容設定爲 ″hms″，否則此內容無效。	否	″:″ (冒號) ″.″ (句點) ″,″ (逗點) ″b″ (空格)	(伺服器工作)

效能內容

效能內容是指影響效能的屬性，其中包括快取、資料轉換、資料壓縮及預先提取。

效能內容	說明	必要的	選項	預設
″大十進位″	指定是否在壓縮及劃分區域十進位轉換使用中階的 java.math.BigDecimal 物件。如果此內容設定爲 ″true″， 則如 JDBC 規格所述，中階的 java.math.BigDecimal 物件會用於壓縮及區化十進位轉換。如果此內容設定爲 ″false″，則沒有中階物件會用於壓縮及區化十進位轉換。而這類的值將直接轉換爲 Java 倍精準度數值，或從 Java 倍精準度數值轉換。這類的轉換會比較快，但可能並未遵循 JDBC 規格所記載的所有轉換及資料截斷規則。	否	″true″ ″false″	″true″
″區塊準則″	指定自伺服器擷取區塊記錄資料的基準。若爲此內容指定非零值，將可減少與伺服器的通訊頻率，因而提升效能。 如果游標將用於後續的更新，請確定已關閉區塊標示記錄，否則所更新的列不需爲目前的列。	否	″0″ (不封鎖任何記錄) ″1″ (如果指定 FOR FETCH ONLY 則封鎖) ″2″ (除非指定 FOR UPDATE，否則封鎖)	″2″
″區塊大小″	指定自伺服器及由用戶端上快取擷取的區塊大小 (以千位元組 (KB) 爲單位)。除非 ″區塊準則″ 內容值不是零，否則此內容無效。較大的區塊大小可降低與伺服器的通訊頻率，因而提升效能。	否	″0″ ″8″ ″16″ ″32″ ″64″ ″128″ ″256″ ″512″	″32″
″資料壓縮″	指定是否壓縮結果集資料。如果此內容設定爲 ″true″，則會壓縮結果集資料。如果此內容設定爲 ″false″，則不壓縮結果集資料。資料壓縮在擷取大量的結果集時，可增進效能。	否	″true″ ″false″	″true″
″延伸動態″	指定是否要使用延伸的動態支援。延伸的動態支援提供伺服器上快取動態 SQL 陳述式的機制。第一次準備某特定 SQL 陳述式時，它會儲存在伺服器上的 SQL 資料包。若資料包不存在，將會自動建立之。往後在準備使用同一個 SQL 陳述式時，伺服器可以使用儲存在 SQL 資料包的資訊，省略掉大部分的處理程序。如果設定爲 ″true″，則必須透過 ″資料包″ 內容來設定資料包名稱。	否	″true″ ″false″	″false″
″延遲關閉″	指定是否要等到後續的要求才延遲關閉游標。如此可減少要求總數，進而提升整體效能。	否	″true″ ″false″	″false″

效能內容	說明	必要的	選項	預設
″LOB 臨界值″	指定可以擷取作爲部分結果集的 LOB (大型物件) 大小上限 (以位元組爲單位)。當 LOB 大於這個臨界值時，將使用伺服器的額外通訊，以片斷方式擷取它們。較大的 LOB 臨界值雖可減少伺服器通訊的頻率， 但會下載更多的 LOB 資料，即使是一些用不到資料。較小的 LOB 臨界值可能增加伺服器通訊的頻率，但它們僅下載需要的 LOB 資料。	否	″0″ - ″16777216″	″32768″
″資料包″	指定 SQL 資料包的基本名稱。 請注意，只會使用前面 7 個字元來產生伺服器上 SQL 資料包的名稱。除非 ″延伸動態″ 內容設定爲 ″true″，否則此內容無效。此外，如果 ″延伸動態″ 內容設定爲 ″true″，則必須設定此內容。	否	SQL 資料包	″″
″資料包新增″	指定是否在 ″資料包″ 內容所指定的 SQL 資料包中，新增最近準備的陳述式。除非 ″延伸動態″ 內容設定爲 ″true″， 否則此內容無效。	否	″true″ ″false″	″true″
″資料包快取″	指定是否快取用戶端記憶體中的 SQL 資料包資訊子集。在本端快取 SQL 資料包減少伺服器在準備及說明上的通訊量。除非 ″延伸動態″ 內容設定爲 ″true″， 否則此內容無效。	否	″true″ ″false″	″false″
″資料包準則″	指定要儲存在 SQL 資料包中的 SQL 陳述式類型。對增進複雜之結合條件的效能而言十分有用。除非 ″延伸動態″ 內容設定爲 ″true″，否則此內容無效。	否	″預設″ (只在資料包中儲存具有參數記號的 SQL 陳述式) ″選取″ (在資料包中儲存所有 SQL SELECT 陳述式)	″預設″
″資料包錯誤″	指定當發生 SQL 資料包錯誤時，要採取的動作。 發生 SQL 資料包錯誤時，驅動程式會根據此內容的值，選擇性地丟出 SQLException 或對「連線」發佈通知。 除非 ″延伸動態″ 內容設定爲 ″true″， 否則此內容無效。	否	″異常″ ″警告″ ″無″	″警告″
″資料包檔案庫″	指定 SQL 資料包的檔案庫。除非 ″延伸動態″ 內容設定爲 ″true″， 否則此內容無效。	否	SQL 資料包的檔案庫	″QGPL″
″預先提取″	指定是否要在執行 SELECT 陳述式時預先提取資料。 這麼做可在存取 ResultSet 中的起始列時提升效能。	否	″true″ ″false″	″true″
″qaqqinilib″	指定 QAQQINI 檔案庫名稱。用來指定要使用的 qaqqini 檔案所在的檔案庫。qaqqini 檔案包含所有可能會影響 DB2 UDB for iSeries 資料庫引擎效能的屬性。	否	″QAQQINI 檔案庫名稱″	(伺服器預設值)

效能內容	說明	必要的	選項	預設
″查詢最佳化目標″	指定伺服器應與最佳化查詢搭配使用的目標。此設定對應於伺服器中名爲 OPTIMIZATION_GOAL 的 QAQQINI 選項。**註:** 連接到執行 i5/OS V5R3 及先前版本的系統時，會忽略此內容。	否	″0″ = 使用延伸動態資料包時最佳化查詢第一個資料區塊 (*FIRSTIO)，不使用資料包時最佳化查詢整個結果集 (*ALLIO) ″1″ = 最佳化查詢第一個資料區塊 (*FIRSTIO) ″2″ = 最佳化查詢整個結果集 (*ALLIO)	″0″

排序內容

排序內容指定伺服器執行儲存及排序的方法。

排序內容	說明	必要的	選項	預設
″排序″	指定在伺服器傳送記錄到用戶端之前，伺服器排序記錄的方法。	否	″十六進位″ (根據十六進位值排序) ″工作″ (根據伺服器工作的設定排序) ″語言″ (根據″排序語言″內容中設定的語言排序) ″表格″ (根據″排序表″內容中設定的排序順序表排序)	″工作″
″排序語言″	指定在進行排序順序的選項時，所要使用的三個字元的語言 ID。除非 ″排序″ 內容設定爲 ″語言″，否則此內容無效。	否	語言 ID	ENU
″排序表″	指定在伺服器中儲存排序順序表的檔案庫及檔名。除非 ″排序″ 內容設定爲 ″表格″，否則此內容無效。	否	限定的排序表格名稱	″″

排序內容	說明	必要的	選項	預設
"排序加權"	指定伺服器在排序記錄時，如何處理情況。除非 "排序" 內容設定爲 "語言"，否則此內容無效。	否	"共用" (大小寫字元排序時視爲相同字元) "唯一" (大小寫字元排序時視爲不同字元)	"共用"

其它內容

其它內容是指不易分類的內容。這些內容判定所使用的 JDBC 驅動程式，並指定與資料庫存取、雙向字串類型及資料截斷等相關的選項。

其它內容	說明	必要的	選項	預設
"存取"	指定連線的資料庫存取層次。	否	"全部" (允許全部 SQL 陳述式) "讀取呼叫" (允許 SELECT 及 CALL 陳述式) "唯讀" (只允許 SELECT 陳述式)	"全部"
"行爲置換"	指定要置換的 IBM Toolbox for Java 驅動程式行爲。您可以在此內容中新增常數並傳送該總和，以變更組合中的多個行爲。請確定您的應用程式會正確地處理變更的行爲。	否	"" (不置換任何行爲) "1" (如果 Statement.executeQuery() 或 PreparedStatement.executeQuery() 沒有傳回結果集，則不會丟出異常，但會傳回空值作爲結果集)	""
"雙向字串類型"	指定雙向資料的輸出字串類型。若需取得更多資訊，請參閱 BidiStringType。	否	"" (使用 CCSID 來決定雙向字串類型) "0" (非雙向資料 (LTR) 的預設字串類型) "4" "5" "6" "7" "8" "9" "10" "11"	""
"雙向隱含重新排序"	指定是否應使用雙向隱含 LTR-RTL 重新排序。	否	"true" "false"	"true"
"雙向數值重新排序"	指定是否應使用數值排序來回轉換功能。	否	"true" "false"	"false"

其它內容	說明	必要的	選項	預設
″資料截斷″	指定**字元資料**截斷時是否產生警告通知和異常。當此內容是 ″true″ 時，則下列情況成立： • 將截斷字元資料寫入資料庫時擲出異常 • 在查詢中使用截斷的字元資料時會發佈警告通知。 當此內容是 ″false″ 時，若將截斷的資料寫入資料庫或在查詢中使用這類資料，將不會產生異常或警告通知。 預設值是 ″true″。 此內容不會影響**數值資料**。將截斷的數值資料寫入資料庫時一律丟出錯誤，在查詢中使用截斷的數值資料時一律發佈警告通知。	否	″true″ ″false″	″true″
″驅動程式″	指定 JDBC 驅動程式實作。IBM Toolbox for Java JDBC 驅動程式可依據環境使用不同的 JDBC 驅動程式實作。如果環境是與程式連接的資料庫在相同伺服器上的 iSeries JVM，則可使用原有的 IBM Developer Kit for Java JDBC 驅動程式。在其他環境中，則使用 IBM Toolbox for Java JDBC 驅動程式。如果設定 ″次要 URL″ 內容，則此內容無效。	否	″工具箱″ (只使用 IBM Toolbox for Java JDBC 驅動程式) ″原有的″ (如果在伺服器上執行，則使用 IBM Developer Kit for Java JDBC 驅動程式，否則就使用 IBM Toolbox for Java JDBC 驅動程式)	″工具箱″
″錯誤″	指定伺服器上發生錯誤時，於訊息中所傳回的明細量。	否	″基本″ ″全部″	″基本″

其它內容	說明	必要的	選項	預設
″延伸的 meta 資料″	指定驅動程式是否向伺服器要求延伸的 meta 資料。 將此內容設定爲 TRUE，可增加從下列 ResultSetMetaData 方法傳回之資訊的精確度： • getColumnLabel(int) • isReadOnly(int) • isSearchable(int) • isWriteable(int) 此外，設定此內容爲 TRUE 會啓用 ResultSetMetaData.getSchemaName(int) 方法的支援。將此內容設定爲 TRUE 可能會降低效能，因爲這麼做需要從伺服器擷取更多資訊。除非您需要由所列出的方法中得到更多特定的資訊，否則請保留此內容的預設值 (false)。 例如，當關閉此內容時 (false)，ResultSetMetaData.isSearchable (int) 會一律傳回「true」，因爲驅動程式並未從伺服器取得可供判斷的足夠資訊。開啓此內容 (true) 會強制驅動程式由伺服器取得正確的資料。 只有在連接到執行 i5/OS V5R2 或更新版本的伺服器時，才能使用延伸 Meta 資料。	否	″true″ ″false″	″false″
″完整開啓″	指定伺服器是否完整開啓每一個查詢的檔案。依據預設值，伺服器會以最佳化開啓要求。這種最佳化會增進效能，但是，當查詢的執行超過一次時，如果資料庫監督程式爲作用中則可能會失敗。因此，僅可在發出相同的查詢且監督程式爲作用中時，將此內容設定爲 TRUE。	否	″true″ ″false″	″false″
″保留輸入定位器″	指定是應將輸入定位器配置爲類型保留定位器，還是非保留定位器。如果定位器爲類型保留定位器，則在確定完成後不會將其釋放。	否	″true″ (類型保留) ″false″	″true″
″保留陳述式″	指定自動確定已關閉且陳述式與 LOB 定位器相關聯時，這些陳述式是否應保持開啓，直到異動界限爲止。依預設，陳述式關閉後，會釋放與陳述式相關聯的所有資源。只有在關閉陳述式之後需要 LOB 定位器時，才應將此內容設爲 true。	否	″true″ ″false″	″false″
″金鑰環名稱″	指定伺服器用於 SSL 連線的金鑰環類別名稱。 除非 ″安全″ 設定爲 TRUE，並且使用 ″金鑰環密碼″ 內容設定金鑰環密碼，否則此內容無效。	否	″金鑰環名稱″	″″

其它內容	說明	必要的	選項	預設
″金鑰環密碼″	指定伺服器用於 SSL 通訊的金鑰環類別密碼。 除非 ″安全″ 設定爲 true，而且使用 ″金鑰環名稱″ 內容設定金鑰環名稱，否則此內容無效。	否	″金鑰環密碼″	″″
″PROXY 伺服器″	指定正在執行 PROXY 伺服器的主電腦名稱及中間層機器的埠。 其格式爲 *hostname*[:*port*]，其中 port 部分是可選用的。若未設定此項，則會從 *com.ibm.as400.access.AS400.proxyServer* 內容擷取主電腦名稱及埠。預設的埠爲 *3470* (如果連線使用 SSL， 則預設的埠爲 *3471*)。必須在中間層機器上執行 ProxyServer。 在二層式環境下，會省略中間層機器的名稱。	否	PROXY 伺服器主電腦名稱及埠	(proxyServer 內容的值，若未設定則沒有值)
″註解″	指定 DatabaseMetaData 方法所傳回 ResultSets 中之 REMARKS 直欄的文字來源。	否	″sql″ (SQL 物件註解) ″系統″ (i5/OS 物件說明)	″系統″
″次要 URL″	如果所指定之多層式環境內中間層 DriverManager 上連線所使用的 URL 與原指定的不同，則在此指定之。此內容可讓您使用此驅動程式連接到 iSeries 伺服器以外的資料庫。在 URL 的反斜線及分號之前，使用反斜線當成跳出字元。	否	JDBC URL	(目前的 JDBC URL)
″安全″	指定是否使用 Secure Sockets Layer (SSL) 連線與伺服器通訊。SSL 連線僅可用於連接 V4R4 或更新版本的伺服器。	否	″true″ (加密所有主從架構通訊) ″false″ (只加密密碼)	″false″
″伺服器追蹤″	指定 JDBC 伺服器工作的追蹤層次。當啓用追蹤時，追蹤會由用戶端連線到伺服器時開始，而在切斷連線時結束。您必須於連線到伺服器之前啓動追蹤，因爲用戶端僅在連線時才啓用伺服器追蹤。	否	″0″ (追蹤未處於作用中) ″2″ (在 JDBC 伺服器工作上啓動資料庫監視程式) ″4″ (在 JDBC 伺服器工作上啓動除錯) ″8″ (當 JDBC 伺服器工作結束時儲存工作日誌) ″16″ (在 JDBC 伺服器工作上啓動工作追蹤) ″32″ (儲存 SQL 資訊) ″64″ (支援啓動資料庫主電腦伺服器追蹤) 將這些值加在一起可啓動多種追蹤類型。例如，″6″ 會啓動資料庫監督程式並啓動除錯。	″0″
″緒的使用″	指定與主電腦伺服器通訊時是否使用緒。	否	″true″ ″false″	″true″

其它內容	說明	必要的	選項	預設
″工具箱追蹤″	指定要記載的 IBM Toolbox for Java 追蹤種類。追蹤訊息在呼叫 JDBC 之程式的除錯上十分有用。不過，記載追蹤訊息會降低效能，因此只有在除錯時，才須設定此內容。追蹤訊息會記錄於 System.out。	否	″″ ″無″ ″資料串流″ (記載本端主電腦與遠端系統之間的資料流) ″診斷″ (記載物件狀態資訊) ″錯誤″ (記載引起異常的錯誤) ″資訊″ (用來透過程式碼來追蹤控制流程) ″警告″ (記載可回復的錯誤) ″轉換″ (記載 Unicode 與原本的字碼頁之間的字集轉換) ″proxy″ (記載用戶端與 proxy 伺服器之間的資料流) ″pcml″ (用來決定 PCML 如何解譯傳入或來自伺服器的資料) ″jdbc″ (記載 jdbc 資訊) ″全部″ (記載所有種類) ″緒″ (記載緒資訊)	″″
″追蹤″	指定是否記載追蹤訊息。追蹤訊息在呼叫 JDBC 之程式的除錯上十分有用。不過，記載追蹤訊息會降低效能，因此只有在除錯時，才須將此內容設定爲 ″true″。追蹤訊息會記錄於 System.out。	否	″true″ ″false″	″false″
″轉換二進位″	指定是否要轉換二進位資料。如果此內容設定爲 ″true″，則 BINARY 欄位和 VARBINARY 欄位會被視爲 CHAR 欄位和 VARCHAR 欄位。	否	″true″ ″false″	″false″

JDBC Librarylist 內容

JDBC LibraryList 內容可指定要新增到伺服器工作檔案庫清單或置換該清單的一個或多個檔案庫， 以及選擇性地設定預設檔案庫 (預設綱目)。

下面表格中的範例做出這些假設：

- 名爲 MYLIBDAW 的檔案庫包含 MYFILE_DAW
- 您將執行此 SQL 陳述式：

```
"SELECT * FROM MYFILE_DAW"
```

實務範例	SQL 命名	系統命名
基本規則	只搜尋一個檔案庫。 • 如果在 URL 中指定某檔案庫，則會使用該檔案庫。它會成爲預設檔案庫。 • 如果在 URL 中未指定檔案庫，則會使用「檔案庫」內容中的第一個檔案庫。它會成爲預設檔案庫。 • 如果 URL 沒有檔案庫而且未指定檔案庫內容，則會使用與登入的使用者設定檔同名的檔案庫。 工作的檔案庫清單會以檔案庫內容中的檔案庫來更新。這樣可能會影響部分觸發程式和儲存程序的行爲。但不影響陳述式中的不完整的名稱。	工作的檔案庫清單會以檔案庫內容中的檔案庫來更新。如果在 URL 中指定某預設檔案庫，則該檔案庫會成爲預設檔案庫。
1. 未指定檔案庫。	預設綱目是使用者設定檔名稱。	無預設綱目。會搜尋工作的檔案庫清單。
2. 已在 URL 中指定預設檔案庫。	預設綱目是指定的檔案庫。	預設綱目是指定的檔案庫。不搜尋檔案庫清單來解析 SQL 陳述式中的不完整名稱。
3. 已透過內容指定預設檔案庫。	預設綱目是指定的檔案庫。	無預設綱目。搜尋清單上所有的檔案庫。
4. 已在 URL 和內容中指定預設檔案庫。	預設綱目是在 URL 中指定的檔案庫。 忽略檔案庫清單。	預設綱目是在 URL 中指定的檔案庫。 不搜尋檔案庫清單來解析 SQL 陳述式中的不完整名稱。
5. 已指定檔案庫內容，檔案庫名稱錯誤	預設綱目是指定的檔案庫	無預設綱目。找不到清單中其中一個檔案庫，因此檔案庫清單無法變更，所以會使用工作的檔案庫清單。
6. URL 沒有檔案庫，檔案庫內容已指定，在清單的第二個檔案庫中找到檔案	預設綱目是清單中的第一個檔案庫，其餘檔案庫會被忽略。	如果所有檔案庫都存在，則沒有預設綱目，會搜尋清單上的所有檔案庫，清單會置換工作的檔案庫清單。 如果清單上其中一個檔案庫不存在，則不變更工作的檔案庫清單。
7. 已指定檔案庫內容，清單的開頭是逗點	預設綱目是使用者設定檔	無預設綱目，搜尋清單上的所有檔案庫，清單置換工作的檔案庫清單。
8. 已指定檔案庫內容，清單的開頭是 *LIBL	預設綱目是使用者設定檔	無預設綱目，搜尋清單上的所有檔案庫，指定的檔案庫新增到清單結尾
9. 已指定檔案庫內容，清單的結尾是 *LIBL	預設綱目是清單上的第一個檔案庫，忽略清單的其餘部分	無預設綱目，搜尋清單上的所有檔案庫，指定的檔案庫新增到工作的檔案庫清單開頭
10. URL 檔案庫無效	無預設綱目，使用使用者設定檔	無預設綱目，使用工作的檔案庫清單

註: 若在 URL 中指定預設綱目，且未使用檔案庫內容，則預設綱目會附加在現行檔案庫清單前面

JDBC SQL 類型

i5/OS 的 DB2 並不支援 JDBC 規格所說明的所有 SQL 類型。

未支援的 SQL 類型

在有不支援的 SQL 類型時，JDBC 驅動程式會以類似的 SQL 類型來取代。

下表列出不支援的 SQL 類型，以及 JDBC 驅動程式用來取代的 SQL 類型。

未支援的 SQL 類型	取代的 SQL 類型
BIT	SMALLINT
TINYINT	SMALLINT
BIGINT (i5/OS 版本 4 版次 4 及之前的版本)	INTEGER
LONGVARCHAR	VARCHAR
LONGVARBINARY	VARBINARY

註: i5/OS V4R5 與以上的版本支援 BIGINT。

Proxy 支援

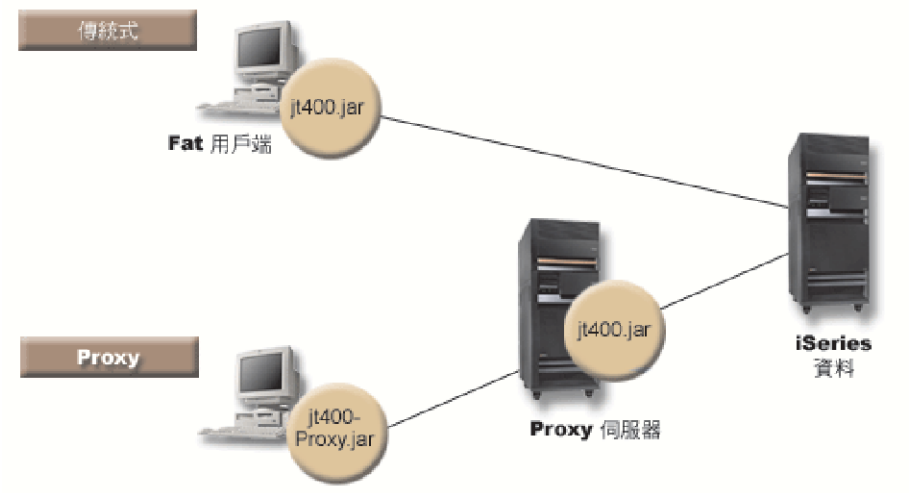
IBM Toolbox for Java 包含部分類別的 proxy 支援。Proxy 支援是一種處理程序，IBM Toolbox for Java 需要用它來在非應用程式實際所在的 Java 虛擬機器 (JVM) 上執行作業。

Proxy 支援包含使用 Secure Sockets Layer (SSL) 通訊協定以加密資料。

proxy 類別位於 jt400Proxy.jar，隨附於 IBM Toolbox for Java 的其餘部分。proxy 類別與 IBM Toolbox for Java 中的其他類別相同，包含一組與平台無關的 Java 類別，可在含有 Java 虛擬機器的任何一部電腦上執行。proxy 類別分派所有的方法呼叫到伺服器應用程式， 或是 PROXY 伺服器。完整的 IBM Toolbox for Java 類別位於 proxy 伺服器上。當用戶端使用 proxy 類別時，此要求會轉送到建立並管理實際 IBM Toolbox for Java 物件的 proxy 伺服器。

圖 1 所示爲標準和 proxy 用戶端連線到伺服器的方式。proxy 伺服器可以是包含資料的 iSeries。

圖 1：標準用戶端和 proxy 用戶端如何連線到伺服器



使用 proxy 支援的應用程式執行速度要比使用標準的 IBM Toolbox for Java 類別來得慢，這是因爲它需要額外的通訊來支援較小的 proxy 類別。執行方法呼叫較少的應用程式，其效能較不會降低。

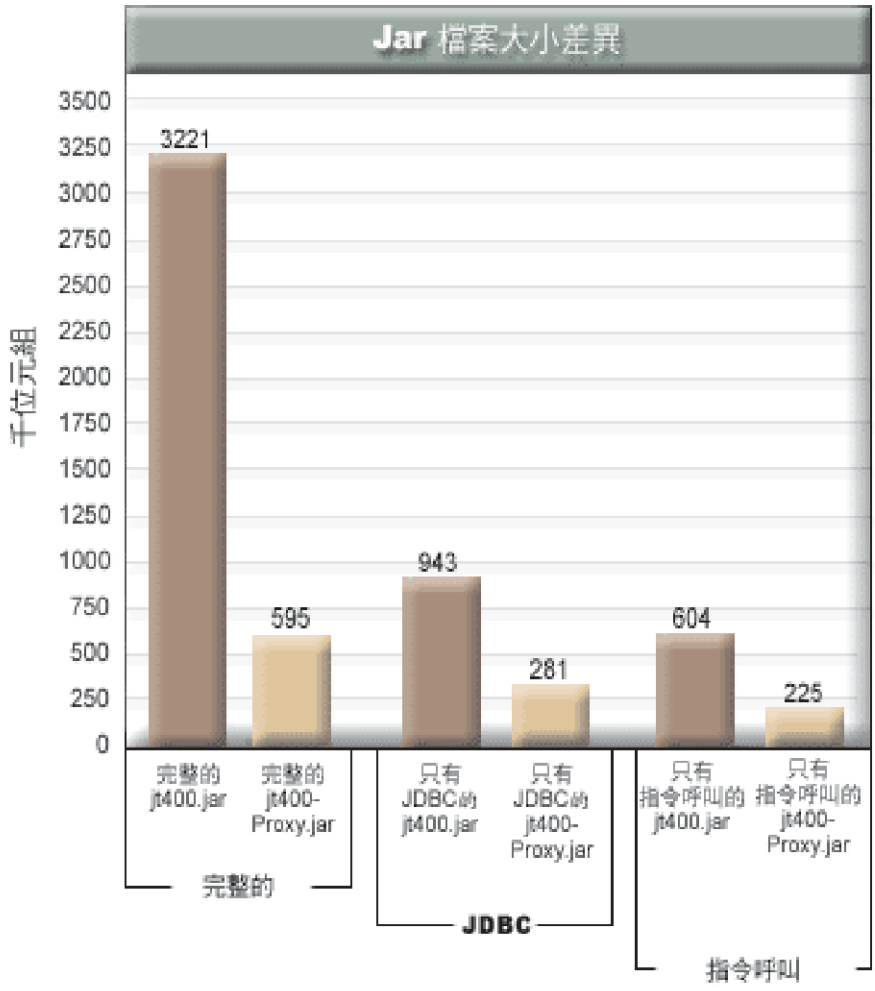
在提供 proxy 支援之前，包含公用介面的類別、所有需要處理要求的類別和應用程式本身都在同一部 Java 虛擬機器上執行。 使用 proxy 支援時，公用介面必須和應用程式在一起，但處理要求用的類別可在另一部 JVM 執行。Proxy 支援並不會變更公用介面。此相同程式可以與 IBM Toolbox for Java 的 proxy 版本或標準版本一起執行。

使用 jt400Proxy.jar 檔案

多重階層的 proxy 實務目標是讓公用介面的 jar 檔案大小儘可能得小，進而使用最少的時間從 Applet 下載此檔。當您使用 proxy 類別時，並不需要在用戶端上安裝完整的 IBM Toolbox for Java。應使用 jt400Proxy.jar 檔案的 AS400JarMaker， 只併入必要的元件，以使 jar 檔案儘可能小。

圖 2 與標準的 jar 檔案比較 proxy jar 檔案之大小：

圖 2：proxy jar 檔案與標準 jar 檔案之間大小的比較



另外還有一個好處就是，使用 proxy 支援時，須透過防火牆開啓的埠較少。使用標準的 IBM Toolbox for Java，您必須開啓多重的埠。這是因爲每項 IBM Toolbox for Java 服務都使用不同的埠與伺服器通訊。例如「指令」呼叫使用非 JDBC 的埠，它使用與列印不同的埠...等等。每一個埠都必須允許它穿過防火牆。 然而，使用 proxy 支援後，所有的資料流程皆經由相同的埠。

標準 proxy 和 HTTP 通道

經由 proxy 執行有兩種選項可用：標準 proxy 和 HTTP 通道：

- 標準 proxy 爲 proxy 用戶端和 proxy 伺服器在埠上使用插座通訊的地方。 預設埠是 3470。 您可以使用 ProxyServer 類別上的 setPort() 方法變更預設的埠，或在啓動 Proxy 伺服器時使用 -port 選項。例如：

```
java com.ibm.as400.access.ProxyServer -port 1234
```

- HTTP 通道爲 proxy 用戶端和 proxy 伺服器在埠上經由 HTTP 伺服器通訊的地方。IBM Toolbox for Java 提供可處理 proxy 要求的 Servlet。Proxy 用戶端經由 HTTP 伺服器呼叫 servlet。通道的優點在於，因爲是經由 HTTP 埠來進行通訊，所以您不必開啓其他透過防火牆的埠。通道的缺點就是比標準 proxy 速度慢。

IBM Toolbox for Java 使用 proxy 伺服器名稱來判斷使用的是標準 proxy 還是通道 proxy：

- 如果是標準 proxy，請使用伺服器名稱。例如：

```
com.ibm.as400.access.AS400.proxyServer=myServer
```

- 如果是通道 proxy，請用 URL 來強制 proxy 用戶端使用通道。例如：

```
com.ibm.as400.access.AS400.proxyServer=http://myServer
```

執行標準 proxy 時，socket 連線存在於用戶端與伺服器之間。如果該連線失敗，伺服器會清除該用戶端的相關資源。

使用 HTTP 通道時，如果使用 HTTP 通訊協定，proxy 無法連線。亦即爲，每一個資料流都會有新的連線。因爲通訊協定無法連線，所以用戶端應用程式不再作用時，伺服器並不知道。結果伺服器也不知道何時應清除資源。通道伺服器則是依照預定的間隔 (基於逾時值) 使用執行緒來清除資源，因此解決了這個問題。

於預定間隔終了時，執行緒就會執行並清除最近未曾使用的資源。有兩個系統內容支配執行緒：

- com.ibm.as400.access.TunnelProxyServer.clientCleanupInterval 爲執行清除執行緒的頻率，以秒計。預設爲每兩小時。
- com.ibm.as400.access.TunnelProxyServer.clientLifetime 爲資源在清除之前可以閒置多久，以秒計。預設是 30 分鐘。

使用 proxy 伺服器

爲了使用 IBM Toolbox for Java 類別的 proxy 伺服器實作，請完成下列步驟：

1. 於 jt400Proxy.jar 上執行 AS400ToolboxJarMaker，捨棄不需要的類別。這是選用步驟，但建議使用。
2. 傳遞 jt400Proxy.jar 到用戶端。若爲 Java Applet，您可能可以從 HTML 伺服器下載 jar 檔。
3. 決定您要用作 PROXY 伺服器的伺服器。
   - 對 Java 應用程式而言，proxy 伺服器可以是任一部電腦。
   - 對 Java Applet 而言，proxy 伺服器必須在與 HTTP 伺服器相同的電腦上執行。
4. 確定已將 jt400.jar 置於伺服器上之 CLASSPATH 中。
5. 啓動 proxy 伺服器或使用 proxy servlet：
   - 若爲標準 proxy，使用下列指令來啓動 proxy 伺服器：

     ```
     java com.ibm.as400.access.ProxyServer
     ```

   - 若爲通道 proxy，請將 HTTP 伺服器配置爲使用 proxy servlet。servlet 類別名稱是 com.ibm.as400.access.TunnelProxyServer， 包含於 jt400.jar 中。
6. 在用戶端上，設定一系統內容以識別 proxy 伺服器。IBM Toolbox for Java 使用此系統內容來判斷所使用的是標準 proxy 還是通道 proxy。
   - 若爲標準 proxy，內容值是執行 proxy 伺服器的機器名稱。例如：

```
com.ibm.as400.access.AS400.proxyServer=myServer
```

- 若爲通道 proxy，請用 URL 來強制 proxy 用戶端使用通道。例如：

```
com.ibm.as400.access.AS400.proxyServer=http://myServer
```

7. 請執行用戶端程式。

如果您都要使用 proxy 類別與不在 jt400Proxy.jar 中的類別兩者， 您可以參照 jt400.jar 以代替 jt400Proxy.jar。jt400Proxy.jar 爲 jt400.jar 的一個子集，因此所有的 proxy 類別都會內含於 jt400.jar 檔案。

使用 SSL

使用 proxy 時，當資料從 proxy 用戶端流向目標 iSeries 伺服器時，有三個選項可以加密資料。SSL 演算法用來加密資料。

1. proxy 用戶端和 proxy 伺服器之間的資料流可以加密。
2. proxy 伺服器與目標 iSeries 伺服器之間的資料流可以加密。
3. 第 1 和第 2 點。proxy 用戶端與 proxy 伺服器之間的資料流，以及 proxy 伺服器與目標 iSeries 之間的資料流可以加密。

請參閱 Secure Sockets Layer 以取得其它相關資訊。

範例：使用 proxy 伺服器

下列三個特定的範例將說明，如何以上述步驟來使用 PROXY 伺服器。

- 使用 proxy 支援執行 Java 應用程式
- 使用 proxy 支援執行 Java Applet
- 使用通道 proxy 支援執行 Java 應用程式

已啓用類別來使用 proxy 伺服器

已啓用部分 IBM Toolbox for Java 類別來使用 proxy 伺服器應用程式。這包括下列各項：

- JDBC
- 記錄層次存取
- 整合檔案系統
- 列印
- 資料佇列
- 指令呼叫
- 程式呼叫
- 服務程式呼叫
- 使用者空間
- 資料區
- AS400 類別
- SecureAS400 類別

到目前爲止，其它類別尚未支援 jt400Proxy。 同時，於僅使用 proxy jar 檔案之下無法使用整合檔案系統許可權。然而，您可以使用 JarMaker 類別， 來含括 jt400.jar 檔案的這些類別。

圖 1 的長說明：標準用戶端和 proxy 用戶端如何連線到伺服器 (rzahh505.gif)

位於 IBM Toolbox for Java：Proxy 支援

此圖說明以下內容：

- 標準用戶端和 proxy 用戶端如何連線到伺服器
- 必要的 IBM Toolbox for Java jar 檔

說明

此圖由下列項目所構成：

- 左上方有個人電腦的影像，代表傳統式的 (標準) 用戶端。這個影像中包括一個圓圈，內含必要的 IBM Toolbox for Java jar 檔名稱 (jt400Proxy.jar)。
- 左下方有個人電腦影像，代表 proxy 用戶端。這個影像中包括一個圓圈，內含必要的 IBM Toolbox for Java jar 檔名稱。
- 中央偏下方有 iSeries 伺服器的影像，代表 proxy 伺服器。這個影像中包括一個圓圈，內含必要的 IBM Toolbox for Java jar 檔名稱。
- 右方有 iSeries 伺服器的影像，代表 iSeries 伺服器。iSeries 伺服器不需要 jar 檔，所以沒有圓圈。
- 連接各個影像的直線。

傳統式 (標準) 用戶端上具有「FAT 用戶端」的標籤，並使用 jt400.jar 檔。標準的用戶端會直接連接到 iSeries 伺服器。

proxy 用戶端使用的是 jt400Proxy.jar 檔。proxy 用戶端會連線到 proxy 伺服器，使用的是 jt400.jar 檔。proxy 伺服器會連接到 iSeries 伺服器。

圖 1 的長說明：proxy jar 檔案與標準 jar 檔案之間大小的比較 (rzahh502.gif)

見 IBM Toolbox for Java：Proxy 支援

這是一張長條圖，將經過使用 AS400JarMaker 來縮減 jar 檔的大小之後的標準與 proxy jar 檔案作出比較。

說明

這張圖表把 jt400.jar 檔對照 jt400Proxy.jar 檔作出下列三種比較：

- 完整的 jar 檔案
- 只包含 JDBC 元件時的 jar 檔案
- 只包含「指令呼叫」元件時的 jar 檔案

這張長條圖由水平軸 (x 軸) 和垂直軸 (y 軸) 所組成：

- 圖表的水平軸 (或 x 軸) 由代表 jar 檔案的三組長條所組成。
- 圖表的垂直軸 (或 y 軸) 代表 jar 檔案以千位元組 (KB) 測量出來的大小。

群組標記成「完整」、JDBC 和「指令呼叫」。每一組各含有 代表 jt400.jar 檔的長條和 jt400Proxy.jar 檔案的長條。各組的說明如下：

- 「完整」：完整的 jt400.jar 檔案大小是 3,221 千位元組 (KB)， 完整的 jt400Proxy.jar 檔案大小則是 595 千位元組 (KB)。

- JDBC：jt400.jar 的檔案大小是 943 千位元組 (KB)，jt400Proxy.jar 的 檔案大小則是 281 千位元組 (KB)，分別只包含 JDBC 元件。
- 「指令呼叫」：jt400.jar 的檔案大小是 604 千位元組 (KB)，jt400Proxy.jar 的檔案大小則是 225 千位元組 (KB)，分別只包含「指令呼叫」元件。

範例：使用 Proxy 支援執行 Java 應用程式

下列範例將說明如何使用 proxy 支援來執行 Java 應用程式的步驟。

1. 選擇要作爲 PROXY 伺服器的機器。proxy 伺服器機器上的 Java 環境及 CLASSPATH 會包含 jt400.jar 檔案。這部機器必須能夠連接 iSeries 伺服器。
2. 若要啓動這部電腦上的 PROXY 伺服器，請鍵入：java com.ibm.as400.access.ProxyServer -verbose，指定 verbose 可讓您監視用戶端連線與斷線的時間。
3. 選擇要作爲用戶端的電腦。用戶端機器上的 Java 環境及 CLASSPATH 會包含 jt400Proxy.jar 檔案及您的應用程式類別。這部機器必須能夠連接到 proxy 伺服器，但不需連接 iSeries 伺服器。
4. 將 com.ibm.as400.access.AS400.proxyServer 系統內容值設定爲 PROXY 伺服器的名稱，然後執行應用程式。有一個簡單的方法可執行這項設定，就是在大部分的「Java 虛擬機器」呼叫中使用 -D 選項：java -Dcom.ibm.as400.access.AS400.proxyServer=psMachineName YourApplication
5. 如果您在步驟 2 設定 verbose，則當應用程式執行時，就會看見此應用程式至少建立了一條連線到 PROXY 伺服器。

範例：使用 proxy 支援執行 Java Applet

下列範例說明如何使用 proxy 支援來執行 Java Applet 的步驟。

1. 選擇要作爲 PROXY 伺服器的機器。Applet 只能起始網路連線到原本下載他們的機器；因此在執行 HTTP 伺服器的機器上 執行 PROXY 伺服器，這種效能最好。proxy 伺服器機器上的 Java 環境及 CLASSPATH 會包含 jt400.jar 檔案。
2. 若要啓動這部電腦上的 PROXY 伺服器，請鍵入：java com.ibm.as400.access.ProxyServer -verbose，指定 verbose 可讓您監視用戶端連線與斷線的時間。
3. Applet 程式碼需要在執行之前下載，所以最好儘可能地減少程式碼大小。若只併入您的 Applet 所使用的元件程式碼，AS400ToolboxJarMaker 將可大幅縮小 jt400Proxy.jar。例如，如果 Applet 僅使用 JDBC，請執行下列指令， 藉以減少 jt400Proxy.jar 檔案，併入最少量的程式碼：

   ```
   java utilities.AS400ToolboxJarMaker -source jt400Proxy.jar -destination jt400ProxySmall.jar
                                       -component JDBC
   ```

4. 此 Applet 必須將 com.ibm.as400.access.AS400.proxyServer 系統內容值設定爲您的 PROXY 伺服器名稱。對於 applet 來說有種簡便的作法：使用已編譯 Properties 類別 (範例)。編譯此類別，然後將產生的 Properties.class 檔案放置在 com/ibm/as400/access 目錄中 (與您的 html 檔案來源路徑相同)。例如， 若 html 檔案是 /mystuff/HelloWorld.html，那麼 Properties.class 就位於 /mystuff/com/ibm/as400/access。
5. 將 jt400ProxySmall.jar 放在與 html 檔案相同的目錄下 (步驟 4 中的 /mystuff/)。
6. 參照此 Applet，如您的 HTML 檔案中所列出的下列字行：

   ```
   <APPLET archive="jt400Proxy.jar, Properties.class" code="YourApplet.class"
           width=300 height=100> </APPLET>
   ```

範例：使用通道 Proxy 支援執行 Java 應用程式

下列範例將說明如何使用通道 proxy 支援來執行 Java 應用程式的步驟。

1. 先選擇您想要執行 proxy 伺服器的 HTTP 伺服器，然後把它配置成執行 servlet com.ibm.as400.access.TunnelProxyServer (用 jt400.jar)。**附註**：請確定 HTTP 伺服器已確實連接到應用程式所使用之資料或資源所在的 iSeries 伺服器，因爲 Servlet 會連接到該 iSeries 以執行要求。

2. 請選擇一部電腦作爲用戶端之用，並確定用戶端電腦中的 CLASSPATH 包含 jt400Proxy.jar 檔案和您的應用程式類別。這個用戶端必須能夠連接到 HTTP 伺服器，但不需連接 iSeries 伺服器。
3. 將 com.ibm.as400.access.AS400.proxyServer 內容值設定爲您 HTTP 伺服器名稱的 URL 格式。
4. 執行應用程式，把 com.ibm.as400.access.AS400.proxyServer 內容值設定爲您 HTTP 伺服器名稱的 URL 格式。有一個簡單的設定方法是在大部分的「Java 虛擬機器」呼叫中使用 -D 選項：

```
java -Dcom.ibm.as400.access.AS400.proxyServer=http://psMachineName YourApplication
```

註: Proxy 用戶端程式碼會把 servlet 和 servlet 名稱連結到伺服器名稱， 藉以建立正確的 servlet URL。在本範例中則是把 http://psMachineName 轉換成 http://psMachineName/servlet/TunnelProxyServer

Secure Sockets Layer 及 Java Secure Socket Extension

IBM Toolbox for Java 支援將 Java Secure Socket Extension (JSSE) 用於 Java Secure Sockets Layer (SSL) 連線。JSSE 是 Java 2 Platform 標準版 (J2SE) 1.2 及 1.3 版的選用資料包。JSSE 已整合到 J2SE 1.4 版及後續版本中。

有關 JSSE 的詳細資訊，請參閱 Sun JSSE 網站 。

JSSE 提供的功能包括：執行伺服器鑑別、啓用安全通訊以及加密資料。使用 JSSE 時，您可以在透過 TCP/IP 執行任何應用程式通訊協定 (例如，HTTP 及 FTP) 的用戶端與伺服器之間，提供安全的資料交換。

如果您之前已使用 SSLight，則應該移轉到 JSSE。在 V5R4 中，JSSE 是受支援的唯一資料包，而不再隨附 SSLight。

IBM Toolbox for Java 2 Micro Edition

IBM Toolbox for Java 2 Micro Edition 套件 (com.ibm.as400.micro) 可讓您撰寫具有下列功能的 Java 程式：允許 Tier0 無線裝置 (如個人數位助理 (PDA) 及行動電話) 直接存取 iSeries 資料及資源。

下載及設定 ToolboxME for iSeries

您必須個別下載 ToolboxME for iSeries (jt400Micro.jar)，其包含於 JTOpen 中。

IBM Toolbox for Java/JTOpen 網站 除了可供您下載 ToolboxME for iSeries，另外還提供設定 ToolboxME for iSeries 的其他資訊。

在 Tier0 裝置、開發工作站及伺服器上設定 ToolboxME for iSeries 的方法不同：

- 爲您的無線裝置建置一個應用程式 (使用 jt400Micro.jar)，並依建置製造商提供的文件中之說明安裝應用程式。
- 請確定 iSeries 主電腦伺服器於包含目標資料的伺服器上啓動。
- 請確定要執行 MEServer 的系統能存取 jt400.jar。

詳細資訊，請參閱下列網頁：

第 10 頁的『在工作站中安裝 IBM Toolbox for Java』

第 9 頁的『在 iSeries 伺服器中安裝 IBM Toolbox for Java』

使用 ToolboxME for iSeries 的重要概念

在開始 ToolboxME for iSeries Java 應用程式的開發之前，您必須瞭解下列支配此項開發作業的概念及標準。

Java 2 Platform, Micro Edition (J2ME)

J2ME 實作 Java 2 標準，提供 Tier0 無線裝置 (如個人數位助理 (PDA) 及行動電話) 的 Java 執行時間環境。IBM Toolbox for Java 2 Micro Edition 遵守此標準。

Tier0 裝置

使用無線技術與電腦及網路連線的無線裝置，如 PDA 及行動電話，稱爲 Tier0 裝置。此名稱的命名方式源自常用的 3 層應用程式模型。3 層模型描述組織成三個主要部分的分散式程式，每一個部分常駐在不同的電腦或網路：

- 第三層爲資料庫及相關程式，通常常駐於與第二層不同的伺服器中。本層提供其它層用來執行作業使用的資訊，及對這些資訊的存取權。
- 第二層是商業邏輯，它通常位於在網路上共用的另一部電腦 (通常爲伺服器)。
- 第一層通常是應用程式部分，常駐在工作站中，包括使用者介面。

Tier0 裝置通常體積小、可攜帶、資源受限，像是 PDA 及行動電話。Tier0 裝置可代替第一層的裝置，或補其功能之不足。

Connected Limited Device Configuration (CLDC)

定義提供與大型裝置相同功能所需之最小 API 集，及必要 Java 虛擬機器功能的配置。CLDC 適用於廣泛的資源受限裝置，包括 Tier0 裝置。

如需相關資訊，請參閱 CLDC 。

行動資訊裝置設定檔 (MIDP)

代表建置在現有配置上、適用於特定類型的裝置或作業系統之 API 集的設定檔。MIDP 建置於 CLDC 之上，提供一標準執行時間環境，可讓您對 Tier0 裝置動態地部署應用程式及服務。

其它相關資訊，請參閱行動資訊 裝置設定檔 (MIDP) 。

用於無線裝置的 Java 虛擬機器

爲了執行 Java 應用程式，您的 Tier0 裝置必須要有特別爲無線裝置之有限資源設計的 Java 虛擬機器。您可使用的 JVM 包括：

- IBM J9 虛擬機器，IBM WebSphere® Micro Environment 的一部分
- Sun K Virtual Machine (KVM), part of CLDC
- MIDP

相關資訊

您可使用任何一種爲協助您建置無線 Java 應用程式而建立的開發工具。有關此種工具的簡略清單，請參閱 IBM Toolbox for Java 的相關資訊。

若要深入瞭解及下載無線裝置模擬器和模擬程式，請至相關網站，查閱您要用來執行應用程式的裝置或作業系統。

ToolboxME for iSeries 類別

com.ibm.as400.micro 資料包可提供撰寫應用程式所需的類別，這些應用程式可讓 Tier0 裝置存取 iSeries 伺服器資料及資源。

com.ibm.as400.micro 資料包

第 316 頁的『Tier0 裝置』

註: 若要使用 ToolboxMe for iSeries 類別，您必須個別下載及設定 ToolboxME for iSeries 元件。

ToolboxME for iSeries 提供下列類別：

- 『MEServer 類別』會將 Tier0 裝置的要求傳送到主電腦伺服器
- 有幾個類別會提供來自 IBM Toolbox for Java存取套件的函數子集
  - 『AS400 類別』會登入 iSeries 伺服器
  - 第 318 頁的『CommandCall 類別』會呼叫 iSeries 指令
  - 第 319 頁的『DataQueue 類別』會讀取及寫入 iSeries 伺服器資料佇列
  - 第 319 頁的『ProgramCall 類別』會呼叫 iSeries 伺服器程式，然後存取程式執行後傳回的資料
- 第 321 頁的『JdbcMe 類別』會併入 java.sql 套件中最小的可用方法與資料集，進而提供 JDBC 支援

MEServer 類別

您可使用 MEServer 類別，滿足來自 Tier0 用戶端應用程式 (使用 ToolboxME for iSeries JAR 檔) 的要求。MEServer 會建立 IBM Toolbox for Java 物件，並代表用戶端應用程式呼叫這些物件中的方法。

註: 若要使用 ToolboxMe for iSeries 類別，您必須個別下載及設定 ToolboxME for iSeries 元件。如需相關資訊，請參閱下載及設定 ToolboxME for iSeries。

您可使用下列指令來啓動 MEServer：

```
java com.ibm.as400.micro.MEServer [options]
```

其中 [options] 爲下列其中一項或數個項目：

-pcml pcml_doc1 [;pcml_doc2;...]
指定 PCML 文件以預載及剖析。您可以使用 -pc來縮寫此選項。

使用此選項的重要相關資訊，請參閱 MEServer javadoc。

-port port
指定用來接受來自用戶端連線的埠。預設埠是 3470。 您可以使用 -po 來縮寫此選項。

-verbose [true|false]
指定是否要將狀態及連線資訊列印至 System.out。您可以使用 -v 來縮寫此選項。

-help 將用法資訊列印至 System.out。您可以使用 -h 或 -? 來縮寫此選項。預設值爲不要列印用法資訊。

如果在指定的埠上已另有一部作用中的伺服器，則 MEServer 將不會啓動。

AS400 類別

micro 套件內的 AS400 類別 (com.ibm.as400.micro.AS400) 提供了存取套件內的 AS400 類別 (com.ibm.as400.access.AS400) 中之可用函數的修正子集。使用 ToolboxMe for iSeries AS400 類別，從 Tier0 裝置登入 iSeries 伺服器。

註: 若要使用 ToolboxMe for iSeries 類別，您必須個別下載及設定 ToolboxME for iSeries 元件。如需相關資訊，請參閱下載及設定 ToolboxME for iSeries。

AS400 類別提供下列功能：

- 連接 MEServer
- 從 MEServer 切斷連接

與 MEServer 的連線是間接地達成。例如，建立 AS400 物件後，便可在 CommandCall 中使用 run() 方法，來自動執行 connect()。換言之，除非您要控制何時建立連線，否則不需明確地呼叫 connect() 方法。

範例：使用 AS400 類別

下列範例將說明如何使用 AS400 類別來登入 iSeries 伺服器：

```
   AS400 system = new AS400("mySystem", "myUserid", "myPwd", "myMEServer");
 try
{
          system.connect();
}
catch (Exception e)
{
           // Handle the exception
}
  // Done with the system object.
  system.disconnect();
```

CommandCall 類別

micro 套件中的 CommandCall 類別 (com.ibm.as400.micro.CommandCall) 提供了存取套件內的 CommandCall 類別 (com.ibm.as400.access.CommandCall) 中之可用函數的修正子集。您可使用 CommandCall 類別，從 Tier0 裝置呼叫 iSeries 指令。

註: 若要使用 ToolboxMe for iSeries 類別，您必須個別下載及設定 ToolboxME for iSeries 元件。

CommandCall run() 方法需要一個 String (您想要執行的指令)，並會在執行 String 之後，傳回產生的所有訊息。如果指令執行完畢後未產生任何訊息，則 run() 方法會傳回空的 String 陣列。

範例：使用 CommandCall

下列範例示範如何使用 CommandCall：

```
  // Work with commands.
  AS400 system = new AS400("mySystem", "myUserid", "myPwd", "myMEServer");
 try
{
    // Run the command "CRTLIB FRED."
    String[] messages = CommandCall.run(system, "CRTLIB FRED");
           if (messages != null)
    {
        // Note that there was an error.
        System.out.println("Command failed:");
        for (int i = 0; i < messages.length; ++i)
        {
            System.out.println(messages[i]);
        }
    }
    else
    {
        System.out.println("Command succeeded!");
    }
```

```
}
catch (Exception e)
{
    // Handle the exception
}
// Done with the system object.
system.disconnect();
```

DataQueue 類別

micro 套件內的 DataQueue 類別 (com.ibm.as400.micro.DataQueue) 提供了存取套件內的 DataQueue 類別 (com.ibm.as400.access.DataQueue) 中之可用函數的修正子集。您可使用 DataQueue 類別讓 Tier0 裝置可讀取或寫入 iSeries 伺服器中的資料佇列。

註: 若要使用 ToolboxMe for iSeries 類別，您必須個別下載及設定 ToolboxME for iSeries 元件。

DataQueue 類別包含下列方法：

- 讀取或撰寫爲 String 的登錄
- 讀取或撰寫爲位元組陣列的登錄

若要讀取或寫入項目，您必須提供資料佇列所在的 iSeries 伺服器名稱，及資料佇列完整的整合檔案系統路徑名稱。沒有可用的項目時，讀取項目會傳回 NULL 值。

範例：使用 DataQueue 來讀取以及寫入資料佇列

下列範例將說明如何使用 DataQueue 類別，在 iSeries 伺服器的資料佇列上讀取以及寫入項目：

```
   AS400 system = new AS400("mySystem", "myUserid", "myPwd", "myMEServer");
 try
{
           // Write to the Data Queue.
    DataQueue.write(system, "/QSYS.LIB/FRED.LIB/MYDTAQ.DTAQ", "some text");

           // Read from the Data Queue.
    String txt = DataQueue.read(system, "/QSYS.LIB/FRED.LIB/MYDTAQ.DTAQ");
}
catch (Exception e)
{
           // Handle the exception
}
  // Done with the system object.
   system.disconnect();
```

ProgramCall 類別

micro 套件內的 ProgramCall 類別 (com.ibm.as400.micro.ProgramCall) 提供了存取套件內的 ProgramCall 類別 (com.ibm.as400.access.ProgramCall) 中之可用函數的已修改子集。您可使用 ProgramCall 類別，讓 Tier0 裝置呼叫 iSeries 程式，並存取執行程式後傳回的資料。

註: 若要使用 ToolboxMe for iSeries 類別，您必須個別下載及設定 ToolboxME for iSeries 元件。如需詳細資訊，請參閱第 7 頁的『ToolboxME for iSeries 基本要求』。

若要使用 ProgramCall.run() 方法，您必須提供下列參數：

- 您要用來執行程式的伺服器
- 第 341 頁的『程式呼叫標記語言』文件的名稱
- 想要執行程式的名稱

- 包含您要設定的一或多個程式參數的名稱及關聯值之雜湊表
- 包含程式執行後將傳回的所有參數之名稱的字串陣列

ProgramCall 使用 PCML 來描述程式的輸入及輸出參數。PCML 檔案所在的機器必須與 MEServer 相同，且您必須在該機器的 CLASSPATH 下 包含 PCML 檔案的目錄。

您必須對 MEServer 註冊每一個 PCML 文件。註冊 PCML 文件的用意就是要告訴 MEServer 您要執行哪一個 PCML 定義的程式。您可在執行期間或啓動 MEServer 時進行 PCML 文件的註冊。

如需包含程式參數之雜湊表或如何註冊 PCML 文件的相關資訊，請參閱 ToolboxME for iSeries ProgramCall javadoc。如需有關 PCML 的詳細資訊， 請參閱第 341 頁的『程式呼叫標記語言』。

範例：使用 ProgramCall

下列範例說明如何使用 ProgramCall 類別來使用 Tier 0 裝置執行伺服器中的程式。

```
    // Call programs.
    AS400 system = new AS400("mySystem", "myUserid", "myPwd", "myMEServer");

    String pcmlName = "qsyrusri.pcml"; // The PCML document describing the program we want to use.
    String apiName = "qsyrusri";

    Hashtable parametersToSet = new Hashtable();
    parametersToSet.put("qsyrusri.receiverLength", "2048");
    parametersToSet.put("qsyrusri.profileName", "JOHNDOE" };

    String[] parametersToGet = { "qsyrusri.receiver.userProfile",
                                 "qsyrusri.receiver.previousSignonDate",
                                 "qsyrusri.receiver.previousSignonTime",
                                 "qsyrusri.receiver.displaySignonInfo" };

String[] valuesToGet = null;

try
{
    valuesToGet = ProgramCall.run(system, pcmlName, apiName, parametersToSet, parametersToGet);

    // Get and display the user profile.
    System.out.println("User profile: " + valuesToGet[0]);

    // Get and display the date in a readable format.
    char[] c = valuesToGet[1].toCharArray();
    System.out.println("Last Signon Date: " + c[3]+c[4]+"/"+c[5]+c[6]+"/"+c[1]+c[2] );

    // Get and display the time in a readable format.
    char[] d = valuesToGet[2].toCharArray();
    System.out.println("Last Signon Time: " + d[0]+d[1]+":"+d[2]+d[3]);

    // Get and display the signon info.
    System.out.println("Signon Info: " + valuesToGet[3] );
}
catch (MEException te)
{
    // Handle the exception.
}
catch (IOException ioe)
{
    // Handle the exception
}

// Done with the system object.
system.disconnect();
```

JdbcMe 類別

ToolboxME for iSeries 類別提供 JDBC 支援，包括支援 java.sql 套件。這些類別主要是設計用來於 Tier 0 裝置中執行的程式內使用。

下節討論資料的存取與使用及說明 JdbcMe 中有什麼，包括單獨 JdbcMe 類別的相關資訊。

資料的存取與使用

使用 Tier0 裝置來存取與更新資料時，您希望裝置的運作情形正如您親自在辦公室的系統前操作一般。但是，大部分 Tier0 裝置的開發著重於資料同步化。藉著使用資料同步化，每一個 Tier0 裝置可具有主資料庫的特定資料之副本。使用者可定期地將裝置上的資料與主資料庫的資料同步化。

動態資料的資料同步化不易實行。將動態資料同步化需要能對最新資料的即時存取。對許多企業而言，不希望在存取同步化資料時要花時間等候。再者，進行資料同步化的伺服器及裝置也需要特殊的硬體與軟體需求。

爲了解決資料同步模型先天上的問題，ToolboxME for iSeries 中的 JdbcMe 類別可讓您對主資料庫執行即時的更新及存取，同時仍可維持離線的資料儲存。因此，您的應用程式可以在針對主資料庫進行即時更新的同時，存取重要的離線資料。此扮演中間角色的方式兼具同步資料模型及即時資料模型的優點。

JdbcMe 中有什麼

根據定義，Tier0 裝置的任何一種驅動程式必須非常小。但是 JDBC API 卻非常大。爲此，JdbcMe 類別必須極小但同時仍能支援足夠的 JDBC 介面，才能讓 Tier0 裝置用來執行有意義的工作。

JdbcMe 類別提供下列 JDBC 功能：

- 插入或更新資料
- 異動控制及修改異動隔離層次的能力
- 同時可捲動及可更新的結果集
- 提供對儲存程序及磁碟機觸發程式呼叫之 SQL 支援

此外，JdbcMe 類別還包含一些唯一的特性：

- 可讓大部分的配置明細於伺服器端的單一點合併之廣用驅動程式
- 讓資料能留存於離線儲存體的標準機制

JdbcMe 包含下列類別：

- JdbcMeConnection
- JdbcMeDriver
- JdbcMeException
- JdbcMeLiveResultSet
- JdbcMeOfflineData
- JdbcMeOfflineResultSet
- JdbcMeResultSetMetaData
- JdbcMeStatement

ToolboxME for iSeries 提供 java.sql 套件，其遵循 JDBC 規格，但僅包含最基本的可用類別及方法集。若提供最小的 SQL 函數集，即可縮小 JdbcMe 類別的大小，但仍足以用來執行常用的 JDBC 作業。

使用 ToolboxME for iSeries 連接主電腦伺服器上的資料庫:

JdbcMeConnection 類別提供了在 IBM Toolbox for Java AS400JDBCConnection 類別中可用的函數子集。您可以使用 JdbcMeConnection 來啓用 Tier0 裝置，以存取主電腦伺服器的 DB2 Universal Database™ (UDB) 資料庫。

註: 若要使用 ToolboxMe for iSeries 類別，您必須個別下載及設定 ToolboxME for iSeries 元件。如需相關資訊，請參閱 ToolboxME for iSeries 基本要求及安裝。

使用 JdbcMeDriver.getConnection() 與伺服器資料庫連接。getConnection() 方法會將全球資源定位器 (URL) 字串視爲引數、 使用者 ID 及密碼。主電腦伺服器上的 JDBC 驅動程式管理員會試圖尋找一驅動程式，該驅動程式可連接至由此 URL 代表的資料庫。JdbcMeDriver 使用下列的 URL 語法：

```
jdbc:as400://server-name/default-schema;meserver=<server>[:port];[other properties];
```

註: 上述語法範例分成兩行，以便您閱讀與列印。正常情況下，URL 會顯示爲連續的一行，不含空格。

您必須指定一個伺服器名稱，否則 JdbcMeDriver 會丟出異常。預設的綱目是可選用的。如果不指定一個埠，JdbcMeDriver 會使用埠 3470。同時，您可在 URL 內設定不同的 JDBC 內容。若要設定內容，請使用下列語法：

```
name1=value1;name2=value2;...
```

請參閱 JDBC 內容，以取得 JdbcMeDriver 支援的內容清單。

範例：使用 JdbcMeDriver 與伺服器連接

範例：在不指定預設綱目、埠或 JDBC 內容的情況下與伺服器資料庫連接

本範例會指定使用者 ID 及密碼作爲方法中的參數：

```
                // Connect to system 'mysystem'. No default schema, port or
        // properties are specified.
  Connection c  = JdbcMeDriver.getConnection("jdbc:as400://mysystem.helloworld.com;meserver=myMeServer;"
                                             "auser",
                                             "apassword");
```

範例：在指定綱目及 JDBC 內容的情況下與伺服器資料庫連接

本範例會指定使用者 ID 及密碼作爲方法中的參數：

```
       // Connect to system 'mysystem'. Specify a schema and
       // two JDBC properties. Do not specify a port.
  Connection c2 = JdbcMeDriver.getConnection(
     "jdbc:as400://mysystem.helloworld.com/mySchema;meserver=myMeServer;naming=system;errors=full;"
     "auser",
     "apassword");
```

範例：連接伺服器資料庫

本範例會使用全球資源定位器 (URL) 來指定內容 (包括使用者 ID 和密碼)：

```
      // Connect using properties. The properties are set on the URL
      // instead of through a properties object.
  Connection c = DriverManager.getConnection(
     "jdbc:as400://mySystem;meserver=myMeServer;naming=sql;errors=full;user=auser;password=apassword");
```

範例：中斷與資料庫的連線

本範例會對連線物件使用 close() 方法，以中斷與伺服器的連線：

```
  c.close();
```

JdbcMeDriver 類別:

JdbcMeDriver 類別提供了 IBM Toolbox for Java AS400JDBCDriver 類別中可用的函數子集。您可在 Tier0 用戶端應用程式中使用 JdbcMeDriver， 來執行沒有參數的簡單 SQL 陳述式，並取得陳述式產生的 ResultSets。

註: 若要使用 ToolboxMe for iSeries 類別，您必須個別下載及設定 ToolboxME for iSeries 元件。如需詳細資訊，請參閱第 315 頁的『下載及設定 ToolboxME for iSeries』。

您並非直接地註冊 JdbcMeDriver；而是於 JdbcMeConnection.getConnection() 方法中 在 URL 指定的 **driver** 來決定驅動程式。例如，若要載入 IBM Developer Kit for Java JDBC 驅動程式 (稱爲「原有的」驅動程式)，請使用類似下列的程式碼：

```
Connection c = JdbcMeDriver.getConnection(
   "jdbc:as400://mysystem.myworld.com;meserver=myMeSrvr;driver=native;user=auser;password=apassword");
```

IBM Toolbox for Java JDBC 驅動程式不像從伺服器取得資料的其他 IBM Toolbox for Java 類別，它不需使用 AS400 物件作爲輸入參數。不過，其會在內部使用 AS400 物件，且您必須直接提供使用者 ID 和密碼。 請在 URL 中提供使用者 ID 和密碼，或以 getConnection() 方法的參數之形式提供使用者 ID 和密碼。

如需使用 getConnection() 的範例，請參閱 JDBCMeConnection。

結果集:

ToolboxME for iSeries 結果集類別爲：

- JdbcMeLiveResultSet
- JdbcMeOfflineResultSet
- JdbcMeResultSetMetaData

JdbcMeLiveResultSet 與 JdbcMeOfflineResultSet 包含相同的功能，除了：

- JdbcMeLiveResultSet 擷取資料的方法是對伺服器的資料庫進行呼叫
- JdbcMeOfflineResultSet 由本機上之資料庫擷取資料

註: 若要使用 ToolboxMe for iSeries 類別，您必須個別下載及設定 ToolboxME for iSeries 元件。如需相關資訊，請參閱下載及設定 ToolboxME for iSeries。

JdbcMeLiveResultSet

JdbcMeLiveResultSet 類別提供了 IBM Toolbox for Java AS400JDBCResultSet 類別中可用的函數子集。您可在 Tier0 用戶端應用程式中使用 JdbcMeLiveResultSet，來存取執行查詢所產生的資料表。

JdbcMeLiveResultSet 會依順序擷取表格列。在一列中，您可以任何次序存取欄位值。JdbcMeLiveResultSet 包含的方法可讓您執行下列作業：

- 擷取儲存於結果集中不同類型的資料
- 移動游標至指定的列 (上一列、現行列、下一列等等)
- 插入、更新及刪除列
- 更新直欄 (使用 String 與 int 值)
- 擷取描述於結果集中直欄的 ResultSetMetaData 物件

游標是一種內部指標，可供結果集用來指向結果集內正由 Java 程式存取的橫列。JDBC 2.0 另外提供一些方法，來存取資料庫內的特定位置：

可捲動的游標位置
absolute
first
last
moveToCurrentRow
moveToInsertRow
previous
relative

捲動功能

如果透過執行一個陳述式來建立結果集，您可以向後 (最後一個到第一個) 或向前 (第一個到最後一個) 移動 (捲動) 表格中的橫列。

支援這個移動的結果集稱爲可捲動的結果集。可捲動的結果集也支援相對與絕對定位。相對位置可讓您經由指定相對於現行列的位置，來移至結果集中的某一列。絕對位置可讓您經由在結果集中指定一個位置，直接移到該橫列。

透過 JDBC 2.0，您有兩個額外捲動功能可在使用 ResultSet 類別時使用：不可捲動及可捲動結果集。

不可捲動結果集通常不會感應到在開啓狀態時所做的變更，而可捲動結果則可感應到變更。IBM Toolbox for Java JDBC 驅動程式不支援不可捲動的結果集。

可更新的結果集

在您的應用程式中，您可以使用使用唯讀並行處理 (不對資料做任何變更) 的結果集， 或可更新並行處理 (容許更新資料並使用資料寫入鎖定來控制不同異動對同一資料的存取權) 的結果集。在可更新結果集中，橫列可被更新、插入及刪除。

請參閱方法總結， 以取得 JdbcMeResultSet 中可用更新方法的完整報表。

範例：可更新的結果集

下面範例將告訴您如何使用當開啓時容許變更資料 (更新並行處理)，以及容許變更結果集 (可捲動) 的結果集。

```
                // Connect to the server.
Connection c = JdbcMeDriver.getConnection(
   "jdbc:as400://mySystem;meserver=myMeServer;user=auser;password=apassword");

     // Create a Statement object.  Set the result set
     // concurrency to updatable.
Statement s = c.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);

     // Run a query. The result is placed
     // in a ResultSet object.
ResultSet rs = s.executeQuery ("SELECT NAME,ID FROM MYLIBRARY.MYTABLE FOR UPDATE");

     // Iterate through the rows of the ResultSet. As we read
     // the row, we will update it with a new ID.
int newId = 0;
while (rs.next ())
{

         // Get the values from the ResultSet. The first value
         // is a string, and the second value is an integer.
    String name = rs.getString("NAME");
    int id = rs.getInt("ID");
```

```
        System.out.println("Name = " + name);
        System.out.println("Old id = " + id);

             // Update the id with a new integer.
        rs.updateInt("ID", ++newId);

             // Send the updates to the server.
        rs.updateRow ();

        System.out.println("New id = " + newId);
    }

         // Close the Statement and the Connection.
    s.close();
    c.close();
```

JdbcMeOfflineResultSet 類別

JdbcMeOfflineResultSet 類別提供了 IBM Toolbox for Java AS400JDBCResultSet 類別中可用的函數子集。您可在 Tier0 用戶端應用程式中 使用 JdbcMeOfflineResultSet，來存取執行查詢所產生的資料表。

您可使用 JdbcMeOfflineResultSet 類別與常駐在 Tier0 裝置的資料搭配使用。常駐在裝置中的資料可能原本就位於該處，或是您呼叫了 JdbcMeStatement.executeToOfflineData() 方法，將資料存放於該處。executeToOfflineData() 方法會下載所有查詢所得的資料，並將它們儲存在裝置中。然後您可使用 JdbcMeOfflineResultSet 類別來存取儲存的資料。

JdbcMeOfflineResultSet 包含的方法可讓您執行下列動作：

- 擷取儲存於結果集中不同類型的資料
- 移動游標至指定的列 (上一列、現行列、下一列等等)
- 插入、更新及刪除列
- 更新直欄 (使用 String 與 int 值)
- 擷取描述於結果集中直欄的 ResultSetMetaData 物件

您可使用 JdbcMe 類別中所示的函數，提供可進行本端裝置資料庫與 iSeries 伺服器之資料庫同步化的功能。

JdbcMeResultSetMetaData 類別

JdbcMeResultSetMetaData 類別提供了在 IBM Toolbox for Java AS400JDBCResultSetMetaData 類別中可用的函數子集。您可在 Tier0 用戶端應用程式中使用 JdbcMeResultSetMetaData，以決定 JdbcMeLiveResultSet 或 JdbcMeOfflineResultSet 中直欄的類型與特性。

下列範例說明如何使用 JdbcMeResultSetMetaData 類別：

```
                // Connect to the server.
    Connection c = JdbcMeDriver.getConnection(
       "jdbc:as400://mySystem;meserver=myMeServer;user=auser;password=apassword");

         // Create a Statement object.
    Statement s = c.createStatement();

         // Run a query. The result is placed in a ResultSet object.
    JdbcMeLiveResultSet rs = s.executeQuery ("SELECT NAME,ID FROM MYLIBRARY.MYTABLE");

         // Iterate through the rows of the ResultSet.
    while (rs.next ())
    {
```

```
            // Get the values from the ResultSet. The first value is
            // a string, and the second value is an integer.
        String name = rs.getString("NAME");
        int id = rs.getInt("ID");

        System.out.println("Name = " + name);
        System.out.println("ID = " + id);
    }

         // Close the Statement and the Connection.
    s.close();
    c.close();
```

JdbcMeOfflineData 類別:

JdbcMeOfflineData 類別爲一離線資料儲存庫，設計使用於 Tier0 裝置上。此儲存庫是通用的，不受您使用的設定檔及 Java 虛擬機器的影響。如需相關資訊，請參閱 ToolboxME for iSeries 概念。

註: 若要使用 ToolboxMe for iSeries 類別，您必須個別下載及設定 ToolboxME for iSeries 元件。如需相關資訊，請參閱下載及設定 ToolboxME for iSeries。

JdbcMeOfflineData 類別提供的方法可讓您執行下列功能：

- 建立離線資料儲存庫
- 開啓現有的儲存庫
- 取得儲存庫中的記錄數
- 取得及刪除個別記錄
- 更新記錄 (Robb: the set() method, right?)
- 新增記錄至儲存庫的尾端
- 關閉儲存庫

如需使用 JdbcMeOfflineData 類別的範例，請參閱下列範例：

第 645 頁的『範例：使用 ToolboxME for iSeries、MIDP 及 IBM Toolbox for Java』

JdbcMeStatement 類別:

JdbcMeStatement 類別提供了 IBM Toolbox for Java AS400JDBCStatement 類別中可用的函數子集。您可在 Tier0 用戶端應用程式中使用 JdbcMeStatement，來執行沒有參數的簡單 SQL 陳述式，及取得陳述式所產生的 ResultSets。

註: 若要使用 ToolboxMe for iSeries 類別，您必須個別下載及設定 ToolboxME for iSeries 元件。如需相關資訊，請參閱下載及設定 ToolboxME for iSeries。

您可使用 JdbcMeConnection.createStatement() 來建立新的 Statement 物件。

下列範例說明如何使用 JdbcMeStatement 物件：

```
                // Connect to the server.
      JdbcMeConnection c = JdbcMeDriver.getConnection(
   "jdbc:as400://mysystem.helloworld.com/mylibrary;naming=system;errors=full;meserver=myMeServer;" +
   "user=auser;password=apassword");

      // Create a Statement object.
  JdbcMeStatement s = c.createStatement();

      // Run an SQL statement that creates a table in the database.
```

```
s.executeUpdate("CREATE TABLE MYLIBRARY.MYTABLE (NAME VARCHAR(20), ID INTEGER)");

     // Run an SQL statement that inserts a record into the table.
s.executeUpdate("INSERT INTO MYLIBRARY.MYTABLE (NAME, ID) VALUES ('DAVE', 123)");

     // Run an SQL statement that inserts a record into the table.
s.executeUpdate("INSERT INTO MYLIBRARY.MYTABLE (NAME, ID) VALUES ('CINDY', 456)");

     // Run an SQL query on the table.
JdbcMeLiveResultSet rs = s.executeQuery("SELECT * FROM MYLIBRARY.MYTABLE");

     // Close the Statement and the Connection.
s.close();
c.close();
```

建立及執行 ToolboxME for iSeries 程式

此資訊可讓您編輯、編譯並執行範例 ToolboxME for iSeries 程式。

您也可將此資訊用來作爲建立、測試及執行 ToolboxME for iSeries 工作範例，及您自己的 ToolboxME for iSeries 應用程式的一般手冊使用。

範例程式使用「K 虛擬機器 (KVM)」，並允許使用者執行所有 JDBC 查詢。接著，使用者可對查詢結果執行 JDBC 動作 (next、previous、close、commit 及 rollback)。

開始建立任何 ToolboxME for iSeries 範例前，請先確定您的環境符合 ToolboxME for iSeries 基本要求。

建立 ToolboxME for iSeries 範例

若要建立 Tier0 裝置的 ToolboxME for iSeries 範例程式，請完成下列步驟：

1. 複製 ToolboxME for iSeries 範例的 Java 程式碼，稱爲 JdbcDemo.java。
2. 在所選的文字或 Java 編輯器中，遵照程式註解的指示變更程式碼，然後以名稱 JdbcDemo.java 儲存檔案。

   註: 請考慮使用無線應用程式開發工具，讓剩餘的步驟更容易完成。某些無線應用程式開發工具可在單一步驟中編譯、預先驗證及建立程式，然後自動在模擬程式中加以執行。
3. 編譯 JdbcDemo.java，請確定是指向包含 KVM 類別的 .jar 檔案。
4. 預先驗證可執行檔，方法是使用無線應用程式開發工具，或使用 Java Preverify 指令。
5. 爲 Tier0 裝置的作業系統建立適當類型的可執行檔。例如，針對 Palm OS，可建立名爲 JdbcDemo.prc 的檔案。
6. 測試程式。如果已安裝了模擬程式，便可在模擬程式中執行程式，以測試程式並看其執行後的情況如何。

   註: 如果您在無線裝置上測試程式時，並未使用無線應用程式開發工具，請確定在裝置中預先安裝所選的 Java 虛擬機器或 MIDP。

請參閱 ToolboxME for iSeries 概念，以取得概念、無線應用程式開發工具及模擬程式的相關資訊。

執行 ToolboxME for iSeries 範例

若要在 Tier0 裝置中執行 ToolboxME for iSeries 範例程式，請完成下列作業：

- 使用 Tier0 裝置製造商所提供的指令，將可執行檔載入裝置。
- 啓動 MEServer
- 按一下 JdbcDemo 圖示，在 Tier0 裝置中執行 JdbcDemo 程式。

ToolboxME for iSeries 範例：JdbcDemo.java

若要建立此範例作爲工作的 ToolboxME for iSeries 程式，您必須複製下列 .java 檔到文字編輯器或 Java 編輯器中，做一些變更，然後加以編譯。

若要複製原始程式碼，只要使用滑鼠選取下面的所有 Java 程式碼，然後按一下滑鼠右鍵並選取**複製**。若要將程式碼貼在編輯器上，請在編輯器中建立一個空白文件，以滑鼠右鍵按一下該空白文件，並選取**貼上**。記得要以 JdbcDemo.java 名稱儲存新的文件。

建立 .java 檔之後，回到指示來建立及執行程式範例。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// ToolboxME for iSeries example. This program demonstrates how your wireless
// device can connect to an iSeries server and use JDBC to perform work on a
// remote database.
//
////////////////////////////////////////////////////////////////////////////

import java.sql.*;               // SQL Interfaces provided by JdbcMe
import com.ibm.as400.micro.*;    // JdbcMe implementation
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.microedition.io.*;  // Part of the CLDC specification
import de.kawt.*;                // Part of the CLDC specification

class DemoConstants
{
    // These constants are actually used mainly by the demo
    // for the JDBC driver. The Jdbc and JDBC  application
    // creator IDs ( http://www.palmos.com/dev )
    // are reserved at palm computing.
    public static final int demoAppID    = 0x4a444243; // JDBC
    // Make the dbCreator something else so that the
    // user can actually see the Palm DB seperately from
    // the JdbcDemo application.
    public static final int dbCreator    = 0x4a444231; // JDB1
    public static final int dbType       = 0x4a444231; // JDB1
}

/**
 * Little configuration dialog box to display the
 * current connections/statements, the
 * URL being used, user id and password
 */
class ConfigurationDialog extends Dialog implements ActionListener
{
    TextField         data;
    ConfigurationDialog(Frame w)
    {
        super(w, "Configuration");

        // Show/Modify current URL connection
        data = new TextField(JdbcDemo.mainFrame.jdbcPanel.url);
        add("Center", data);

        // Ok button.
        Panel panel = new Panel();
        Button button = new Button("Ok");
        button.addActionListener(this);
        panel.add(button);
        add("South", panel);
```

```
        pack();
    }

    public void actionPerformed(ActionEvent e)
    {
        JdbcDemo.mainFrame.jdbcPanel.url = data.getText();
        data = null;
        setVisible(false);
    }
}

/**
 * Little configuration dialog box to display the
 * current connections/statements, the
 * URL being used, user id and password
 */
class MultiChoiceDialog extends Dialog implements ActionListener
{
    Choice            task;
    ActionListener    theListener;
    MultiChoiceDialog(Frame w, String title, String prompt, String choices[], ActionListener it)
    {
        super(w, title);
        theListener = it;

        // Show/Modify current URL connection
        Label txt = new Label(prompt);
        add("West", txt);
        task = new Choice();
        for (int i=0; i<choices.length; ++i)
        {
            task.add(choices[i]);
        }
        task.select(0);
        add("Center", task);

        // Ok button.
        Panel panel = new Panel();
        Button button = new Button("Ok");
        button.addActionListener(this);
        panel.add(button);
        button = new Button("Cancel");
        button.addActionListener(this);
        panel.add(button);
        add("South", panel);
        pack();
    }

    /**
     *  Determine the action performed.
     **/
    public void actionPerformed(ActionEvent e)
    {
        int choice = task.getSelectedIndex();
        setVisible(false);
        if (e.getActionCommand().equals("Ok"))
        {
            if (theListener != null)
            {
                ActionEvent ev = new ActionEvent(this,
                                                 ActionEvent.ACTION_PERFORMED,
                                                 task.getItem(choice));
                theListener.actionPerformed(ev);
            }
            task = null;
        }
        else
```

```
        {
            // No-op
        }
    }
}

/**
 * The JdbcPanel is the main panel of the application.
 * It displays the current connection and statement
 * at the top.
 * A text field for entering SQL statements next.
 * A Results field for displaying each column of data
 * or results.
 * An task list and a 'go' button so that different
 * tasks can be tried.
 */
class JdbcPanel extends Panel implements ActionListener
{
    public final static int TASK_EXIT         =  0;
    public final static int TASK_NEW          =  1;
    public final static int TASK_CLOSE        =  2;
    public final static int TASK_EXECUTE      =  3;
    public final static int TASK_PREV         =  4;
    public final static int TASK_NEXT         =  5;
    public final static int TASK_CONFIG       =  6;
    public final static int TASK_TOPALMDB     =  7;
    public final static int TASK_FROMPALMDB   =  8;
    public final static int TASK_SETAUTOCOMMIT=  9;
    public final static int TASK_SETISOLATION = 10;
    public final static int TASK_COMMIT       = 11;
    public final static int TASK_ROLLBACK     = 12;


    // JDBC objects.
    java.sql.Connection  connObject  = null;
    Statement            stmtObject  = null;
    ResultSet            rs          = null;
    ResultSetMetaData    rsmd        = null;

    String        lastErr     = null;
    String        url         = null;
    Label         connection  = null;
    Label         statement   = null;
    TextField     sql         = null;
    List          data        = null;
    final Choice  task;

    /**
     * Build the GUI.
     */
    public JdbcPanel()
    {
        // The JDBC URL
        // Make sure to edit the following line so that it correctly specifies the
        // the MEServer and the iSeries server to which you want to connect.
        url = "jdbc:as400://mySystem;user=myUidl;password=myPwd;meserver=myMEServer;";

        Panel    p1left = new Panel();
        p1left.setLayout(new BorderLayout());
        connection = new Label("None");
        p1left.add("West", new Label("Conn:"));
        p1left.add("Center", connection);

        Panel    p1right = new Panel();
        p1right.setLayout(new BorderLayout());
        statement = new Label("None");
        p1right.add("West", new Label("Stmt:"));
```

```
    p1right.add("Center", statement);

    Panel    p1 = new Panel();
    p1.setLayout(new GridLayout(1,2));
    p1.add(p1left);
    p1.add(p1right);

    Panel    p2 = new Panel();
    p2.setLayout(new BorderLayout());
    p2.add("North", new Label("Sql:"));
    sql = new TextField(25);
    sql.setText("select * from QIWS.QCUSTCDT"); // Default query
    p2.add("Center", sql);

    Panel    p3 = new Panel();
    p3.setLayout(new BorderLayout());
    data = new List();
    data.add("No Results");
    p3.add("North", new Label("Results:"));
    p3.add("Center", data);

    Panel    p4 = new Panel();

    task = new Choice();
    task.add("Exit");             // TASK_EXIT
    task.add("New");              // TASK_NEW
    task.add("Close");            // TASK_CLOSE
    task.add("Execute");          // TASK_EXECUTE
    task.add("Prev");             // TASK_PREV
    task.add("Next");             // TASK_NEXT
    task.add("Config");           // TASK_CONFIGURE
    task.add("RS to PalmDB");     // TASK_TOPALMDB
    task.add("Query PalmDB");     // TASK_FROMPALMDB
    task.add("Set AutoCommit");   // TASK_SETAUTOCOMMIT
    task.add("Set Isolation");    // TASK_SETISOLATION
    task.add("Commit");           // TASK_COMMIT
    task.add("Rollback");         // TASK_ROLLBACK
    task.select(TASK_EXECUTE);    // Start off here.
    p4.add("West", task);

    Button b = new Button("Go");
    b.addActionListener(this);
    p4.add("East", b);

    Panel prest = new Panel();
    prest.setLayout(new BorderLayout());
    prest.add("North", p2);
    prest.add("Center", p3);
    Panel pall = new Panel();
    pall.setLayout(new BorderLayout());
    pall.add("North", p1);
    pall.add("Center", prest);

    setLayout(new BorderLayout());
    add("Center", pall);
    add("South", p4);
}

/**
 * Do a task based on whichever task is
 * currently selected in the task list.
 */
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() instanceof MultiChoiceDialog)
    {
        String   cmd = e.getActionCommand();
```

```
            processExtendedCommand(cmd);
            return;
        }

        switch (task.getSelectedIndex())
        {
        case TASK_EXIT:
            System.exit(0);
            break;
        case TASK_NEW:
            JdbcPanel.this.goNewItems();
            break;
        case TASK_PREV:
            JdbcPanel.this.goPrevRow();
            break;
        case TASK_NEXT:
            JdbcPanel.this.goNextRow();
            break;
        case TASK_EXECUTE:
            if (connObject == null || stmtObject == null)
                JdbcPanel.this.goNewItems();

            JdbcPanel.this.goExecute();
            break;
        case TASK_CONFIG:
            JdbcPanel.this.goConfigure();
            break;
        case TASK_CLOSE:
            JdbcPanel.this.goClose();
            break;
        case TASK_TOPALMDB:
            if (connObject == null || stmtObject == null)
                JdbcPanel.this.goNewItems();

            JdbcPanel.this.goResultsToPalmDB();
            break;
        case TASK_FROMPALMDB:
            JdbcPanel.this.goQueryFromPalmDB();
            break;
        case TASK_SETAUTOCOMMIT:
            JdbcPanel.this.goSetAutocommit();
            break;
        case TASK_SETISOLATION:
            JdbcPanel.this.goSetIsolation();
            break;
        case TASK_COMMIT:
            JdbcPanel.this.goTransact(true);
            break;
        case TASK_ROLLBACK:
            JdbcPanel.this.goTransact(false);
            break;

        default :
        {
                Dialog dialog = new FeedbackDialog(JdbcDemo.mainFrame, "Error", "Task not implemented");
                dialog.show();
                dialog = null;
            }
        }
    }

    public void processExtendedCommand(String cmd)
    {
        try
        {
            if (cmd.equals("true"))
            {
```

```
            connObject.setAutoCommit(true);
            return;
        }
        if (cmd.equals("false"))
        {
            connObject.setAutoCommit(false);
            return;
        }
        if (cmd.equals("read uncommitted"))
        {
            connObject.setTransactionIsolation(java.sql.Connection.TRANSACTION_READ_UNCOMMITTED);
            return;
        }
        if (cmd.equals("read committed"))
        {
            connObject.setTransactionIsolation(java.sql.Connection.TRANSACTION_READ_COMMITTED);
            return;
        }
        if (cmd.equals("repeatable read"))
        {
            connObject.setTransactionIsolation(java.sql.Connection.TRANSACTION_REPEATABLE_READ);
            return;
        }
        if (cmd.equals("serializable"))
        {
            connObject.setTransactionIsolation(java.sql.Connection.TRANSACTION_SERIALIZABLE);
            return;
        }
        throw new IllegalArgumentException("Invalid command: " + cmd);
    }
    catch (Exception e)
    {
        JdbcDemo.mainFrame.exceptionFeedback(e);
        return;
    }
}

/**
 * Perform commit or rollback processing.
 */
public void goTransact(boolean commit)
{
    if (connObject == null)
    {
        FeedbackDialog dialog = new FeedbackDialog(JdbcDemo.mainFrame,
                                                   "Skip",
                                                   "Connection not allocated");
        dialog.show();
        dialog = null;
        return;
    }
    try
    {
        if (commit)
            connObject.commit();
        else
            connObject.rollback();
    }
    catch (Exception e)
    {
        JdbcDemo.mainFrame.exceptionFeedback(e);
    }
}

/**
 * Prompt the user for setting the autocommit value
 * Real work handled by the actionPerformed method
```

```
 * calling processExtendedCommand().
 */
public void goSetAutocommit()
{
    if (connObject == null)
    {
        FeedbackDialog dialog = new FeedbackDialog(JdbcDemo.mainFrame,
                                                   "Skip",
                                                   "Connection not allocated");
        dialog.show();
        dialog = null;
        return;
    }
    try
    {
        String currentValue;
        if (connObject.getAutoCommit())
            currentValue = "Now: true";
        else
            currentValue = "Now: false";

        Dialog dialog = new MultiChoiceDialog(JdbcDemo.mainFrame,
                                              "Set Autocommit",
                                              currentValue,
                                              new String[]{ "true", "false"},
                                              this);
        dialog.show();
        dialog = null;
    }
    catch (Exception e)
    {
        JdbcDemo.mainFrame.exceptionFeedback(e);
    }
}

/**
 * Prompt the user for setting the isolation level,
 * real work handled by the actionPerformed() method
 * calling processExtendedCommand().
 */
public void goSetIsolation()
{
    if (connObject == null)
    {
        FeedbackDialog dialog = new FeedbackDialog(JdbcDemo.mainFrame,
                                                   "Skip",
                                                   "Connection not allocated");
        dialog.show();
        dialog = null;
        return;
    }
    try
    {
        int   level = connObject.getTransactionIsolation();
        String currentLevel;
        switch (level)
        {
        case java.sql.Connection.TRANSACTION_READ_UNCOMMITTED:
            currentLevel = "Now: read uncommitted";
            break;
        case java.sql.Connection.TRANSACTION_READ_COMMITTED:
            currentLevel = "Now: read committed";
            break;
        case java.sql.Connection.TRANSACTION_REPEATABLE_READ:
            currentLevel = "Now: repeatable read";
            break;
        case java.sql.Connection.TRANSACTION_SERIALIZABLE:
```

```
            currentLevel = "Now: serializable";
            break;
        default : {
                currentLevel = "error";
            }
        }
        Dialog dialog = new MultiChoiceDialog(JdbcDemo.mainFrame,
                                              "Set Isolation Level",
                                              currentLevel,
                                              new String[]{ "read uncommitted",
                                                            "read committed",
                                                            "repeatable read",
                                                            "serializable"},
                                              this);
        dialog.show();
        dialog = null;
    }
    catch (Exception e)
    {
        JdbcDemo.mainFrame.exceptionFeedback(e);
    }
}

/**
 * Create a new connection or statement.
 * Only one connection and statement is currently
 * supported.
 */
public void goNewItems()
{
    if (connObject != null || stmtObject != null)
    {
        FeedbackDialog dialog = new FeedbackDialog(JdbcDemo.mainFrame,
                                                   "Skip",
                                                   "Conn/Stmt already allocated");
        dialog.show();
        dialog = null;
    }
    if (connObject == null)
    {
        try
        {
            connObject = DriverManager.getConnection(url);
            //connection.setText(Integer.toString(((JdbcMeConnection)connObject).getId()));
            connection.repaint();
        }
        catch (Exception e)
        {
            JdbcDemo.mainFrame.exceptionFeedback(e);
            return;
        }
    }
    if (stmtObject == null)
    {
        try
        {
            try
            {
                stmtObject = connObject.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                                        ResultSet.CONCUR_READ_ONLY);
            }
            catch (Exception e)
            {
                // Try again...  DB2 NT version 6.1 doesn't support
                // Scollable result sets, so we'll assume other
                // JDBC 2.0 databases don't either. We'll attempt
                // to create another.
```

```
                try
                {
                    stmtObject = connObject.createStatement();
                }
                catch (Exception ex)
                {
                    // If the second try failed, rethrow the
                    // first exception. Its probably
                    // a more meaninful error.
                    throw e;
                }
                FeedbackDialog dialog = new FeedbackDialog(JdbcDemo.mainFrame,
                                                           "2nd try worked",
                                                           "Non-scrollable result set");
                dialog.show();
                dialog = null;
            }

            statement.repaint();
        }
        catch (Exception e)
        {
            JdbcDemo.mainFrame.exceptionFeedback(e);
            return;
        }
    }
}


/**
 *  Close the statement and connection.
 **/
public void goClose()
{
    // Close the statement.
    if (stmtObject != null)
    {
        if (rs != null)
        {
            try
            {
                rs.close();
            }
            catch (Exception e)
            {
            }
            rs = null;
            rsmd = null;
        }
        try
        {
            stmtObject.close();
        }
        catch (Exception e)
        {
        }
        stmtObject = null;
        statement.setText("None");
        statement.repaint();
    }

    // Clost the connection.
    if (connObject != null)
    {
        try
        {
            connObject.close();
```

```
        }
        catch (Exception e)
        {
        }
        connObject = null;
        connection.setText("None");
        connection.repaint();
    }
    data.removeAll();
    data.add("No Results");
    data.repaint();
    sql.repaint();
    return;
}

/**
 * display the configuration dialog.
 **/
public void goConfigure()
{
    // Note there is no model dialog support in KAWT, this only
    // works because the data to be changed (url) is set before
    // this dialog is used, and the user cannot access the
    // main frame while this is up on the palm (i.e. all dialogs
    // in Kawt are modal).
    ConfigurationDialog dialog = new ConfigurationDialog(JdbcDemo.mainFrame);
    dialog.show();
    dialog = null;
}

/**
 *  Execute the specified query.
 **/
public void goExecute()
{
    // Get the currently selected statement.
    try
    {
        if (rs != null)
            rs.close();

        rs = null;
        rsmd = null;
        boolean results = stmtObject.execute(sql.getText());
        if (results)
        {
            rs = stmtObject.getResultSet();
            rsmd = rs.getMetaData();
            // Show the first row
            goNextRow();
        }
        else
        {
            data.removeAll();
            data.add(stmtObject.getUpdateCount() + " rows updated");
            data.repaint();
        }
    }
    catch (Exception e)
    {
        JdbcDemo.mainFrame.exceptionFeedback(e);
    }
}


/**
 *  Move to the next row in the result set.
```

```
 **/
public void goNextRow()
{
    try
    {
        if (rs == null || rsmd == null)
            return;

        int   count = rsmd.getColumnCount();
        int   i;
        data.removeAll();
        if (!rs.next())
            data.add("End of data");
        else
        {
            for (i=1; i>=count; ++i)
            {
                data.add(rs.getString(i));
            }
        }
        data.repaint();
    }
    catch (Exception e)
    {
        JdbcDemo.mainFrame.exceptionFeedback(e);
    }
}


/**
 *  Move to the previous row in the result set.
 **/
public void goPrevRow()
{
    try
    {
        if (rs == null || rsmd == null)
            return;

        int   count = rsmd.getColumnCount();
        int   i;
        data.removeAll();
        if (!rs.previous())
            data.add("Start of data");
        else
        {
            for (i=1; i<=count; ++i)
            {
                data.add(rs.getString(i));
            }
        }
        data.repaint();
    }
    catch (Exception e)
    {
        JdbcDemo.mainFrame.exceptionFeedback(e);
    }
}

/**
 *  Perform a query and store the results in the local devices database
 **/
public void goResultsToPalmDB()
{
    try
    {
        if (stmtObject == null)
```

```
            {
                FeedbackDialog dialog = new FeedbackDialog(JdbcDemo.mainFrame, "Skip", "No Statement");
                dialog.show();
                dialog = null;
                return;
            }

            boolean results =
               ((JdbcMeStatement)stmtObject).executeToOfflineData(sql.getText(),
                                                                  "JdbcResultSet",
                                                                  DemoConstants.dbCreator,
                                                                  DemoConstants.dbType);
            if (!results)
            {
                FeedbackDialog dialog = new FeedbackDialog(JdbcDemo.mainFrame, "No Data", "Not a query");
                dialog.show();
                dialog = null;
                return;
            }
            data.removeAll();
            data.add("Updated Palm DB 'JdbcResultSet'");
            data.repaint();
        }
        catch (Exception e)
        {
            JdbcDemo.mainFrame.exceptionFeedback(e);
        }
    }


    /**
     *  Perform a query from the database that resides on the palm device.
     **/
    public void goQueryFromPalmDB()
    {
        try
        {
            if (rs != null)
            {
                rs.close();
                rs = null;
            }
            rs = new JdbcMeOfflineResultSet ("JdbcResultSet",
                                             DemoConstants.dbCreator,
                                             DemoConstants.dbType);
            rsmd = rs.getMetaData();
            // If we want to debug some output, this
            // method can be used to dump the contents
            // of the PalmDB represented by the result set
            // (Uses System.out so its mostly useful in
            // the Palm emulator when debugging your
            // applications.
            // ((JdbcMeOfflineResultSet)rs).dumpDB(true);

            // show the first row.
            goNextRow();
        }
        catch (SQLException e)
        {
            JdbcDemo.mainFrame.exceptionFeedback(e);
        }
    }
}

public class JdbcDemo extends Frame
{
    /** An ActionListener that ends the application. Only
```

```
 * one is required, and can be reused
 */
private static ActionListener    exitActionListener = null;
/**
 * The main application in this process.
 */
static         JdbcDemo mainFrame = null;

JdbcPanel      jdbcPanel = null;

public static ActionListener getExitActionListener()
{
    if (exitActionListener == null)
    {
        exitActionListener = new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                System.exit(0);
            }
        };
    }
    return exitActionListener;
}

/**
 *  Demo Constructor
 **/
public JdbcDemo()
{
    super("Jdbc Demo");
    setLayout(new BorderLayout());

    jdbcPanel = new JdbcPanel();
    add("Center", jdbcPanel);

    addWindowListener(new WindowAdapter()
     {
         public void windowClosing(WindowEvent e)
         {
             System.exit(0);
         }
     });
    setSize(200,300);
    pack();
}

public void exceptionFeedback(Exception e)
{
    Dialog dialog = new FeedbackDialog(JdbcDemo.mainFrame, e);
    dialog.show();
    dialog = null;
}

/**
 * Main method.
 **/
public static void main(String args[])
{
    try
    {
        mainFrame = new JdbcDemo();
        mainFrame.show();
        mainFrame.jdbcPanel.goConfigure();
    }
    catch (Exception e)
    {
```

```
            System.exit(1);
        }
    }
}
```

ToolboxME for iSeries 工作範例

下列 ToolboxMe for iSeries 工作範例，說明 ToolboxME for iSeries 與「行動資訊裝置設定檔 (MIDP)」搭配使用的方法。

您可使用下列鏈結來檢視所選的範例來源檔，或下載所有建置工作範例無線應用程式所需的範例來源檔：

第 637 頁的『範例：使用 ToolboxME for iSeries、MIDP 及 JDBC』

第 645 頁的『範例：使用 ToolboxME for iSeries、MIDP 及 IBM Toolbox for Java』

『下載 ToolboxME for iSeries 範例』

若需如何建置 ToolboxME for iSeries 應用程式的相關資訊，請參閱第 327 頁的『建立及執行 ToolboxME for iSeries 程式』。

如需 MIDP 的相關資訊，請參閱第 316 頁的『行動資訊裝置設定檔 (MIDP)』。

下載 ToolboxME for iSeries 範例

若要在運作中的無線應用程式中建置 ToolboxME for iSeries 範例，必須具備所有的來源檔及附加指令。

若要下載及建立範例，請完成下列步驟：

1. 下載來源檔 (microsamples.zip)。
2. 將 microsamples.zip 解壓縮至建立用來存放該檔案的目錄。
3. 參考第 327 頁的『建立及執行 ToolboxME for iSeries 程式』中所提供的指示，來建立範例無線應用程式。

開始編譯原始檔並爲 Tier0 裝置建置可執行檔之前，請先參閱下列章節以取得相關資訊：

- 第 7 頁的『ToolboxME for iSeries 基本要求』
- 第 315 頁的『下載及設定 ToolboxME for iSeries』

可延伸標記語言元件

IBM Toolbox for Java 包含了數個「可延伸標記語言 (XML)」元件，包括 XML 剖析器。

XML 元件讓您更易於執行各種工作：

- 建立圖形式使用者介面
- 呼叫您的 iSeries 伺服器上的程式並擷取結果
- 指定您的 iSeries 伺服器上的資料格式

程式呼叫標記語言

「程式呼叫標記語言 (PCML)」爲一種標示語言，可協助您使用較少的 Java 程式碼來呼叫伺服器程式。

PCML 是以「可擴充標示語言 (XML)」爲基礎，這是一種您可用來說明伺服器程式的輸入及輸出參數的標示語法。PCML 可讓您定義完整說明 Java 應用程式所呼叫之伺服器程式的標示。

註: 如果您對 PCML 有興趣或已經在使用，則可以考慮使用「可延伸的程式呼叫標記語言(XPCML)」。XPCML 能提供對 XML 綱目的支援， 因此加強了 PCML 的功能性及實用性。如需 IBM Toolbox for Java XML 元件的相關資訊，包括 XPCML，請參閱可延伸標記語言元件。

PCML 的最大好處就是它可讓您少寫些程式。通常，在伺服器及 IBM Toolbox for Java 物件間需要額外的程式碼來連接、擷取及轉換資料。然而，藉由使用 PCML，可自動處理以 IBM Toolbox for Java 類別對伺服器進行的呼叫。PCML 類別物件是從 PCML 標示產生的，可協助將爲了從應用程式呼叫伺服器程式所須撰寫的程式碼數量減至最小。

雖然根據設計，PCML 應支援從用戶端 Java 平台對伺服器程式物件的分散式程式呼叫，但您也可以使用 PCML 從伺服器環境內呼叫伺服器程式。

請參閱下列網頁， 以取得有關 PCML 的詳細資訊：

- 『使用 PCML 建置 iSeries 程式呼叫』
- 第 346 頁的『PCML 語法』
- 第 560 頁的『範例：程式呼叫標記語言 (PCML)』

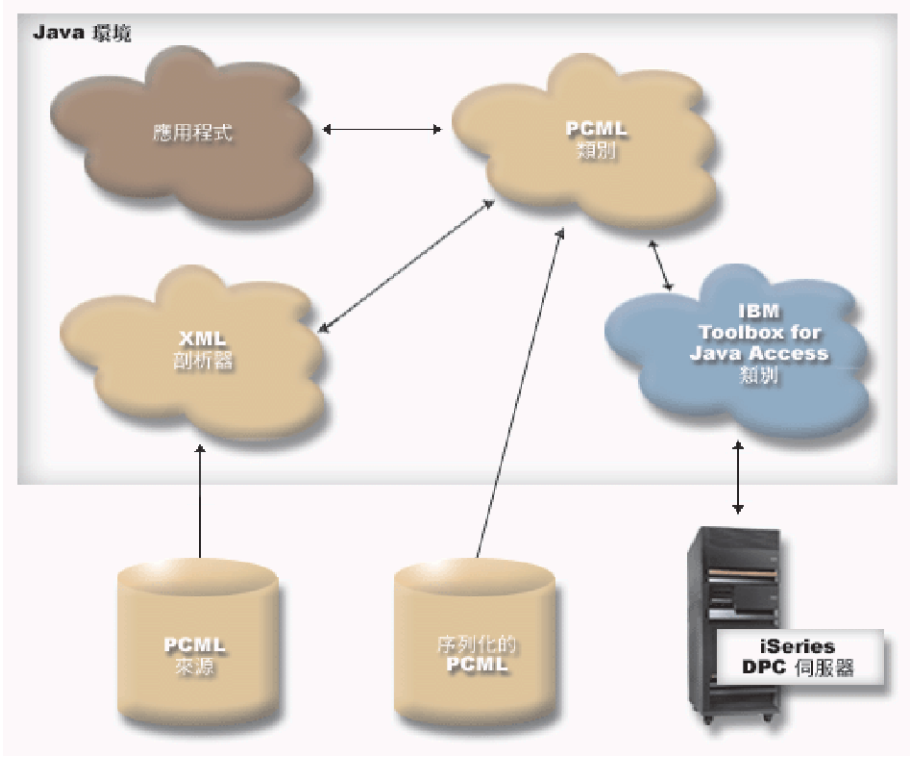
使用 PCML 建置 iSeries 程式呼叫

若要透過 PCML 來建置 iSeries 程式呼叫，首先必須建立 Java 應用程式及 PCML 來源檔。

視不同的設計程序而定，您必須撰寫一或多個 PCML 來源檔，描述您的 Java 應用程式將呼叫的 iSeries 程式介面。請參閱 PCML 語法，取得語言的詳細說明。

然後您的 Java 應用程式必須與 PCML 類別 (本案例中是 ProgramCallDocument 類別) 互動。ProgramCallDocument 類別會利用您的 PCML 來源檔在您的 Java 應用程式及 iSeries 程式之間傳遞資訊。圖 1 說明 Java 應用程式如何與 PCML 類別互動。

圖 1，使用 PCML 產生伺服器的程式呼叫。



當您的應用程式建構 ProgramCallDocument 物件時，XML 剖析器將讀取及解析 PCML 原始檔。若需如何搭配使用 XML 剖析器與 IBM Toolbox for Java 的相關資訊，請參閱 XML 剖析器與 XSLT 處理器。

於建立 ProgramCallDocument 類別後，應用程式將使用 ProgramCallDocument 類別的方法，透過 iSeries 分散式程式呼叫 (DPC) 伺服器，從伺服器擷取必需的資訊。

若要改善執行時間效能，您可以在建置產品時，將 ProgramCallDocument 類別序列化。然後，使用序列化的檔案，建構 ProgramCallDocument。 在這種情況中，不會在執行時間使用 XML 剖析器。請參閱使用序列化 PCML 檔。

使用 PCML 原始檔

| 您的 Java 應用程式會以對 PCML 來源檔的參照建構 ProgramCallDocument 物件，以使用
| PCML。ProgramCallDocument 物件會將 PCML 來源檔視爲 Java 資源。Java 應用程式會使用 Java CLASSPATH
| 來尋找 PCML 來源檔

下列 Java 程式碼會建構 ProgramCallDocument 物件：

```
AS400 as400 = new AS400();
ProgramCallDocument pcmlDoc = new ProgramCallDocument(as400, "myPcmlDoc");
```

ProgramCallDocument 物件會在稱爲 myPcmlDoc.pcml 的檔案中尋找您的 PCML 來源。 請注意，.pcml 副檔名未指定於建構子中。

如果您於 Java 套件中開發 Java 應用程式，可以用套件限定 PCML 資源的名稱：

```
AS400 as400 = new AS400();
ProgramCallDocument pcmlDoc = new ProgramCallDocument(as400, "com.company.package.myPcmlDoc");
```

使用序列化的 PCML 檔

若要改善執行時間效能，您可以使用序列化的 PCML 檔。序列化的 PCML 檔含有代表 PCML 的序列化 Java 物件。序列化的物件同於當您從上述的來源檔 建構 ProgramCallDocument 時所建立的物件。

使用序列化的 PCML 檔將會改善效能，因爲在執行時， 不需要 XML 剖析器來處理 PCML 標籤。

您可以使用下列其中一種方法，將 PCML 序列化：

- 從指令行：

  ```
  java com.ibm.as400.data.ProgramCallDocument -serialize mypcml
  ```

  這個方法有助於以批次處理方式來建置您的應用程式。

- 從 Java 程式內：

  ```
  ProgramCallDocument pcmlDoc; // Initialized elsewhere
  pcmlDoc.serialize();
  ```

如果您 PCML 位於名爲 myDoc.pcml 的原始檔中，序列化的結果即是一個名爲 myDoc.pcml.ser 的檔案。

PCML 來源 vs. 序列化 PCML 檔

考慮使用下列程式來建構 ProgramCallDocument：

```
AS400 as400 = new AS400();
ProgramCallDocument pcmlDoc = new ProgramCallDocument(as400, "com.mycompany.mypackage.myPcmlDoc");
```

ProgramCallDocument 建構子首先會嘗試在 Java CLASSPATH 中的 com.mycompany.mypackage 套件內尋找名爲 myPcmlDoc.pcml.ser 的序列化 PCML 檔。如果序列化的 PCML 檔不存在，則建構子將嘗試在 Java

CLASSPATH 中的 com.mycompany.mypackage 套件內尋找名爲 myPcmlDoc.pcml 的 PCML 來源檔。如果 PCML 來源不存在，將丟出一個異常情況。

完整名稱

您的 Java 應用程式會使用 ProgramCallDocument.setValue() 來設定所要呼叫之 iSeries 程式的輸入值。同樣地，您的應用程式會使用 ProgramCallDocument.getValue() 來擷取 iSeries 程式的輸出值。

在存取 ProgramCallDocument 類別中的值時， 您必須指定文件元素或 <**data**> 標籤的完整名稱。完整名稱即是所有含有的標籤的名稱的組，每一個名稱之間均是以句點來區隔。

例如，以下列 PCML 來源爲例， ″**nbrPolygons**″ 項目的完整名稱爲″**polytest.parm1.nbrPolygons**″。存取多邊形某一邊的某一點的 ″**x**″ 值的完整名稱爲 ″**polytest.parm1.polygon.point.x**″。

如果產生完整名稱所需的任一元素尚未命名，則該元素的所有衍生元素將沒有完整名稱。沒有完整名稱的元素將無法從 Java 程式中存取。

```
<pcml version="1.0">
  <program name="polytest" path="/QSYS.lib/MYLIB.lib/POLYTEST.pgm">
    <!-- Parameter 1 contains a count of polygons along with an array of polygons -->
    <struct name="parm1" usage="inputoutput">
      <data name="nbrPolygons" type="int" length="4" init="5" />
      <!-- Each polygon contains a count of the number of points along with an array of points -->
      <struct name="polygon" count="nbrPolygons">
        <data name="nbrPoints" type="int" length="4" init="3" />
        <struct name="point" count="nbrPoints" >
          <data name="x" type="int" length="4" init="100" />
          <data name="y" type="int" length="4" init="200" />
        </struct>
      </struct>
    </struct>
  </program>
</pcml>
```

存取陣列中的資料

任何 <**data**> 或 <**struct**> 元素都可以使用 **count** 屬性來定義爲陣列。或者，<**data**> 或 <**struct**> 元素可以包含在定義爲陣列的另一個 <**struct**> 元素內。

此外，若有多個含有元素的陣列指定了 **count** 屬性，則 <**data**> 或 <**struct**> 元素就可位於多維度的陣列中。

爲了讓您的應用程式能夠設定或取得定義爲陣列或在陣列內定義的值，您必須指定陣列的每一維度的陣列索引。陣列索引將以 **int** 值的陣列形式傳遞。以上述顯示之多邊形陣列的來源爲例，下列 Java 程式碼可用來擷取多邊形的相關資訊：

```
        ProgramCallDocument polytest; // Initialized elsewhere
        Integer nbrPolygons, nbrPoints, pointX, pointY;
    nbrPolygons = (Integer) polytest.getValue("polytest.parm1.nbrPolygons");
    System.out.println("Number of polygons:" + nbrPolygons);
        indices = new int[2];
        for (int polygon = 0; polygon < nbrPolygons.intValue(); polygon++)
    {
                indices[0] = polygon;
        nbrPoints = (Integer) polytest.getValue("polytest.parm1.polygon.nbrPoints", indices );
        System.out.println("  Number of points:" + nbrPoints);

                for (int point = 0; point < nbrPoints.intValue(); point++)
        {
            indices[1] = point;
            pointX = (Integer) polytest.getValue("polytest.parm1.polygon.point.x", indices );
```

```
            pointY = (Integer) polytest.getValue("polytest.parm1.polygon.point.y", indices );
            System.out.println("    X:" + pointX + " Y:" + pointY);
        }
    }
```

除錯

當您使用 PCML 呼叫具有複雜的資料結構的程式時，容易在您的 PCML 產生錯誤， 導致 ProgramCallDocument 類別發生異常情況。如果錯誤是因爲不正確地描述資料的位移與長度而引起的，將很難對異常情況進行除錯。

請使用 Trace 類別的下列方法來開啓 PCML 追蹤：

```
Trace.setTraceOn(true);      // Turn on tracing function.
Trace.setTracePCMLOn(true);  // Turn on PCML tracing.
```

註: PcmlMessageLog 類別中所有的公用方法 (包括追蹤)， 在 V5R2 中都是停用的。

追蹤 setFileName() 方法可讓您將下列類型的資訊傳送到特定的日誌檔，或按照預設傳送到 System.out：

- Java 應用程式與 iSeries 程式之間轉送的十六進位資料傾出。在字元資料轉換爲 EBCDIC 及整數轉換爲 big-endian 後，這將顯示程式輸入參數。在輸出參數轉換爲 Java 環境之前，它也會顯示這些輸出參數。

  資料會以一般十六進位傾出格式 (十六進位數字在左邊，而字元解譯在右邊) 來顯示。以下就是這種傾出格式的範例：(下列範例已經改變，允許使用寬度限制)

  ```
  qgyolobj[6]
  Offset : 0....... 4....... 8....... C....... 0....... 4....... 8....... C.......
           0...4...8...C...0...4...8...C...
       0 : 5CE4E2D9 D7D9C640 4040
           **USRPRF                        *
  ```

  在上述範例中，傾出顯示第 7 個參數具有設定爲 "*USRPRF" 的資料，共有 10 個位元組。

- 對於輸出參數，在十六進位傾出後是文件的資料如何解譯的說明。(下列範例已經改變，允許使用寬度限制)

```
/QSYS.lib/QGY.lib/QGYOLOBJ.pgm[2]
Offset : 0....... 4....... 8....... C....... 0....... 4....... 8....... C.......
         0...4...8...C...0...4...8...C...
     0 : 0000000A 0000000A 00000001 00000068 D7F0F9F9 F0F1F1F5 F1F4F2F6 F2F5F400
         *................P09901151426254.*
    20 : 00000410 00000001 00000000 00000000 00000000 00000000 00000000 00000000
         *.............................*
    40 : 00000000 00000000 00000000 00000000
         *................                *
Reading data -- Offset: 0   Length: 4   Name: "qgyolobj.listInfo.totalRcds"
 Byte data: 0000000A
Reading data -- Offset: 4   Length: 4   Name: "qgyolobj.listInfo.rcdsReturned"
 Byte data: 0000000A
Reading data -- Offset: 8   Length: 4   Name: "qgyolobj.listInfo.rqsHandle"
 Byte data: 00000001
Reading data -- Offset: c   Length: 4   Name: "qgyolobj.listInfo.rcdLength"
 Byte data: 00000068
Reading data -- Offset: 10  Length: 1   Name: "qgyolobj.listInfo.infoComplete"
 Byte data: D7
Reading data -- Offset: 11  Length: 7   Name: "qgyolobj.listInfo.dateCreated"
 Byte data: F0F9F9F0F1F1F5
Reading data -- Offset: 18  Length: 6   Name: "qgyolobj.listInfo.timeCreated"
 Byte data: F1F4F2F6F2F5
Reading data -- Offset: 1e  Length: 1   Name: "qgyolobj.listInfo.listStatus"
 Byte data: F4
Reading data -- Offset: 1f  Length: 1   Name: "qgyolobj.listInfo.[8]"
 Byte data: 00
Reading data -- Offset: 20  Length: 4   Name: "qgyolobj.listInfo.lengthOfInfo"
 Byte data: 00000410
Reading data -- Offset: 24  Length: 4   Name: "qgyolobj.listInfo.firstRecord"
```

```
 Byte data: 00000001
Reading data -- Offset: 28  Length: 40  Name: "qgyolobj.listInfo.[11]"
 位元組資料：0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

當來自 iSeries 程式的輸出資料與 PCML 來源不符時，上述訊息可能非常有助於診斷。當您使用動態長度與位移時，很容易發生這種情況。

圖 1 的長說明：使用 PCML 對伺服器作程式呼叫 (rzahh503.gif):

位於 IBM Toolbox for Java：PCML 處理

此影像說明 Java 應用程式如何與 PCML 類別互動。

說明

此影像分爲兩個區域：上半部代表 Java 環境，下半部代表 PCML 處理的非 Java 部分。

- Java 環境 (上半部) 包含標示爲「應用程式」、「PCML 類別」、「IBM Toolbox for Java Access 類別」及「XML 剖析器」的四個形狀。
- 處理的非 Java 部分 (下半部) 包含標示爲「PCML 來源」及「PCML 序列化」的兩個形狀，以及 iSeries 伺服器的影像，標示爲 iSeries DPC Server。
- 形狀由單向或雙向箭頭連接起來。有箭頭指向兩個形狀即表示這兩個形狀彼此互動。箭頭如果只指向一個方向表示所指向的形狀以某種方式使用另一個形狀。

Java 應用程式會與 PCML 類別互動。本範例中，應用程式建立 ProgramCallDocument 物件。

當建構 ProgramCallDocument 時，會有以下兩種情形之一發生：

- 由 XML 剖析器剖析 PCML 原始檔，並把資訊傳遞到 PCML 類別。PCML 來源檔說明您 Java 應用程式呼叫的 iSeries 程式介面。
- 會有序列化的 PCML 資訊傳遞到 PCML 類別。由於 PCML 已經過剖析，所以使用序列化的 PCML 可增進執行時間的效能。如果您選擇把 PCML 來源序列化，必須在建置應用程式的時候完成。

PCML 類別也會與 IBM Toolbox for Java 類別互動，在本範例中是使用 iSeries 分散式程式呼叫伺服器來從伺服器擷取資訊。

這些動作及互動，讓資訊能夠在 Java 應用程式與 iSeries 程式之間傳遞。

PCML 語法

PCML 是由下列標記所組成，每一個均有自己的屬性標記。

- program 標記開始並結束描述程式的程式碼。
- struct 標記定義一個已命名的結構，它可指定爲程式的引數，或另一個已命名結構內的欄位。結構標籤含有結構中每一欄位的資料或結構標籤。
- data 標記定義程式或結構內的欄位。

以下範例中，PCML 語法描述一個具有某類別的資料及一些隔離的資料的程式。

```
<program>

    <struct>
        <data> </data>
    </struct>
```

```
    <data> </data>

</program>
```

PCML 程式標籤:

PCML 程式標籤可以下列元素展開。

```
<program name="name"
    [ entrypoint="entry-point-name" ]
    [ epccsid="ccsid" ]
    [ path="path-name" ]
    [ parseorder="name-list" ]
    [ returnvalue="{ void | integer }" ]
    [ threadsafe="{ true | false }" ]>
</program>
```

以下表格列出程式標籤的屬性。每個項目都包括屬性名稱、可能的有效值和屬性的說明。

屬性	值	說明
entrypoint=	*entry-point-name*	在此程式呼叫的目標服務程式物件中，指定進入點的名稱。
epccsid=	*ccsid*	指定服務程式中，進入點的 CCSID。詳細資訊，請參閱 ServiceProgramCall javadoc 中有關服務程式項目的附註。
name=	*name*	指定程式的名稱。
path=	*path-name*	指定程式物件的路徑。預設值假設程式位在 QSYS 檔案庫中。 路徑必須是指向 *PGM 或 *SRVPGM 物件的有效整合檔案系統路徑名稱。如果 *SRVPGM 物件被呼叫， 則必須指定 entrypoint 屬性，以表示被呼叫的 entrypoint 的名稱。 如果沒有指定 entrypoint 屬性，則此屬性的預設值會假設是 QSYS 檔案庫中的 *PGM 物件。 如果已指定 entrypoint 屬性，則此屬性的預設值會假設是 QSYS 檔案庫中的 *SRVPGM 物件。 路徑名稱必須全部以大寫字元指定。 如果應用程式需要在執行時間設定路徑 (例如，使用者指定安裝所用的檔案庫時)，請不要使用 **path** 屬性。在這種情形下，請使用 ProgramCallDocument.setPath() 方法。

屬性	值	說明
parseorder=	*name-list*	指定輸出參數的處理次序。指定的值是以空白區隔的參數名稱清單，將按排列次序處理這些參數。清單中的名稱必須與 **<program>** 所屬標籤的 **name** 屬性中指定的名稱相同。預設值將按標籤出現在文件的次序來處理輸出參數。 某些程式會在參數傳回資訊，描述先前參數中的資訊。例如，假設程式在第一個參數傳回結構陣列，而在第二個參數傳回陣列中的登錄數。在這種情況中，第二個參數必須按 ProgramCallDocument 中的次序處理，來決定第一個參數中要處理的結構數目。
returnvalue=	*void* 程式不會傳回值。 *integer* 程式會傳回 4 位元組帶正負號的整數。	指定服務程式呼叫所傳回的值的類型，如果有的話。這個屬性不可用於 *PGM 物件呼叫。
threadsafe=	*true* 程式將視爲安全緒。 *false* 程式不是安全緒。	當您呼叫同一部伺服器中的 Java 程式以及 iSeries 程式時，請用這個屬性來指定要不要在 Java 程式的同一工作中及同一個緒上呼叫 iSeries 程式。如果確知 程式是安全緒，把內容設定爲 *true* 可使效能更佳。 爲維持環境的安全，預設爲在另外的伺服器工作中呼叫程式。預設值是 *false*。

PCML struct 標籤:

PCML struct 標籤可以下列元素展開。

```
<struct name="name"
    [ count="{number | data-name }"]
    [ maxvrm="version-string" ]
    [ minvrm="version-string" ]
    [ offset="{number | data-name }" ]
    [ offsetfrom="{number | data-name | struct-name }" ]
    [ outputsize="{number | data-name }" ]
    [ usage="{ inherit | input | output | inputoutput }" ]>
</struct>
```

以下表格列出 struct 標籤的屬性。每個項目都包括屬性名稱、可能的有效值和屬性的說明。

屬性	值	說明
name=	*name*	指定 **<struct>** 元素的名稱。

屬性	值	說明
count=	*number*，其中 *number* 定義固定、永不變更大小的陣列。 *data-name*， 其中 *data-name* 會定義 PCML 文件中 **<data>** 元素的名稱，在執行時該名稱將含有陣列中的元素數目。指定的 *data-name* 可以是完整名稱，或是現行元素的相對名稱。不管是哪一種情況，名稱都必須參照以 **type=***″int″* 定義的 **<data>** 元素。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。	指定元素是一個陣列，且識別陣列中的登錄數目。 如果略過這個屬性，則元素雖然不會定義爲陣列，但它可能包含在定義爲陣列的另一個元素內。
maxvrm=	*version-string*	指定該元素所在之處的最高 i5/OS 版本。如果 i5/OS 版本高於屬性上指定的版本，則在呼叫程式時，將不會處理元素及其子項 (如果有的話)。 **maxvrm** 元素有助於定義 i5/OS 版次間不同的程式介面。 版本字串的語法必須是 ″VvRrMm″，其中大寫字母 ″V″、″R″ 及 ″M″ 是文字字元，而 ″v″、″r″ 及 ″m″ 則是代表版本、 版次及修正層次的數字 (一或多位數)。″v″ 的值必須是從 1 到 255。″r″ 及 ″m″ 的值必須是從 0 到 255。
minvrm=	*version-string*	指定此元素所在之處的最低 i5/OS 版本。如果 i5/OS 版本低於此屬性上指定的版本，則在呼叫程式時，將不會處理此元素及其子項 (如果有的話)。此屬性有助於定義 i5/OS 版次間不同的程式介面。 版本字串的語法必須是 ″VvRrMm″，其中大寫字母 ″V″、″R″ 及 ″M″ 是文字字元，而 ″v″、″r″ 及 ″m″ 則是代表版本、 版次及修正層次的數字 (一或多位數)。″v″ 的值必須是從 1 到 255。″r″ 及 ″m″ 的值必須是從 0 到 255。

屬性	值	說明
offset=	*number*，其中 *number* 定義固定、永不變更的偏移。 *data-name*，其中 *data-name* 會定義 PCML 文件中，將在執行時含有此 **<data>** 元素的偏移之元素名稱。指定的 **data-name** 可以是完整名稱，或是現行元素的相對名稱。不管是哪一種情況，名稱都必須參照以 **type=***"int"* 定義的 **<data>** 元素。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。	指定輸出參數內的 **<struct>** 元素的偏移。 有些程式會傳回具有固定結構的資訊，其後跟著一個或多個可變長度欄位或結構。在這種情況下，可變長度元素的位置，通常指定爲參數內的偏移或位移。**offset** 屬性會用來描述這個 **<struct>** 元素的偏移。 **Offset** 將結合 **offsetfrom** 屬性一起使用。 如果未指定 **offsetfrom** 屬性，則 **offset** 屬性上指定的位移的基本位置將是元素的母項。請參閱指定偏移， 以取得如何使用 **offset** 與 **offsetfrom** 屬性的詳細資訊。 **offset** 與 **offsetfrom** 屬性僅會用來處理程式的輸出資料。這些屬性不會控制輸入資料的位移或位置。 如果略過屬性，則元素的資料位置之後緊跟著參數中的前一個元素，若有的話。
offsetfrom=	*number*，其中 *number* 定義固定、永不變更的基本位置。*number* 屬性 最常用來指定 **number=***"0"*，表示位移是從參數開頭開始的絕對位移。 *data-name*，其中 *data-name* 會定義 **<data>** 元素的名稱，用來作爲偏移的基本位置。指定的元素名稱必須是這個元素的母項或祖先項。來自 **offset** 屬性的值，將相對於此屬性上指定元素的位置。指定的 *data-name* 可以是完整名稱，或是現行元素的相對名稱。不管哪一種情況，名稱必須參照這個元素的祖先項。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。 *struct-name*，其中 *struct-name* 會定義 **<struct>** 元素的名稱，用來作爲偏移的基本位置。指定的元素名稱必須是這個元素的母項或祖先項。來自 **offset** 屬性的值，將相對於此屬性上指定元素的位置。指定的 *struct-name* 可爲完整的名稱或是與現行元素相對的名稱。不管哪一種情況，名稱必須參照這個元素的祖先項。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。	指定與**位移**屬性相對的基本位置。 如果未指定 **offsetfrom** 屬性，則 **offset** 屬性上指定的偏移基本位置即爲這個元素的母項。請參閱指定偏移， 以取得如何使用 **offset** 與 **offsetfrom** 屬性的詳細資訊。 **offset** 與 **offsetfrom** 屬性僅會用來處理程式的輸出資料。這些屬性不會控制輸入資料的位移或位置。

屬性	值	說明
outputsize=	*number*，其中 *number* 會定義要保留的固定、永不變更的位元組數目。 *data-name*， 其中 *data-name* 會定義 PCML 文件中 **<data>** 元素的名稱，在執行時該名稱將含有要爲輸出資料保留的位元組數。指定的 *data-name* 可以是完整名稱，或是現行元素的相對名稱。不管是哪一種情況，名稱都必須參照以 **type=**″*int*″ 定義的 **<data>** 元素。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。	指定要保留給元素的輸出資料的位元組數目。對長度可變的輸出參數而言，必須要有 **outputsize** 屬性，才能指定須保留多少個位元組讓將要從伺服器程式傳回的資料使用。**Outputsize** 可以在所有可變長度欄位及可變大小陣列上指定，或它可以針對含有一個或多個可變長度欄位的整個參數而指定。 沒有必要針對固定大小輸出參數指定 **Outputsize**，就算有必要也不得指定。 屬性上指定的值將作爲元素的總大小，包括元素的所有子項。因此，在元素的子項或後代項上，不會處理 **outputsize** 屬性。 如果省略該屬性，則要保留給輸出資料的位元組數，將在爲 **<struct>** 元素的所有子項新增要保留的位元組數時決定。
usage=	*inherit*	用法將繼承自母項元素。如果結構沒有母項，則用法將假設爲 **inputoutput**。
	input	結構是主程式的輸入值。對於字元與數字類型，將執行適當的轉換。
	output	結構是主程式的輸出值。對於字元與數字類型，將執行適當的轉換。
	inputoutput	結構同時爲輸入及輸出值。

指定位移

有些程式會傳回具有固定結構的資訊，其後跟著一個或多個可變長度欄位或結構。在這種情況下，可變長度元素的位置，通常指定爲參數內的偏移或位移。

位移即是從參數開頭到欄位或結構開頭的距離，以位元組表示。位置即是從某個結構的開頭到另一結構開頭的距離，以位元組表示。

對於偏移，因爲距離是從參數開頭開始算起，所以請指定 **offsetfrom=**″**0**″。下列是從參數開頭開始算起的位移的範例：

```
<pcml version="1.0">
  <program name="myprog" path="/QSYS.lib/MYLIB.lib/MYPROG.pgm">
    <!-- receiver variable contains a path -->
    <struct name="receiver" usage="output" outputsize="2048">
      <data name="pathType"         type="int"  length="4" />
      <data name="offsetToPathName" type="int"  length="4" />
      <data name="lengthOfPathName" type="int"  length="4" />
      <data name="pathName"         type="char" length="lengthOfPathName"
                offset="offsetToPathName"  offsetfrom="0"/>
    </struct>
  </program>
</pcml>
```

對於位置，因爲距離是從另一個結構開始算起，所以您可以指定與位移相對的結構的名稱。下列是從已命名的結構開頭開始算起的位置的範例：

```
<pcml ="1.0">
  <program name="myprog" path="/QSYS.lib/MYLIB.lib/MYPROG.pgm">
    <!-- receiver variable contains an object -->
    <struct name="receiver" usage="output" >
      <data name="objectName"       type="char"  length="10" />
      <data name="libraryName"      type="char"  length="10" />
      <data name="objectType"       type="char"  length="10" />
      <struct name="pathInfo" usage="output" outputsize="2048" >
        <data name="pathType"         type="int"  length="4" />
        <data name="offsetToPathName" type="int"  length="4" />
        <data name="lengthOfPathName" type="int"  length="4" />
        <data name="pathName"         type="char" length="lengthOfPathName"
                 offset="offsetToPathName"  offsetfrom="pathInfo"/>
      </struct>
    </struct>
  </program>
</pcml>
```

PCML 資料標記:

PCML 資料標記具有下列屬性。

若以方括弧 ([]) 括住，即代表選用性的屬性。如果您指定了選用性屬性，請不要在您的原始程式中使用方括弧 ([])。有些屬性值會顯示在選擇清單中，以大括弧 ({}) 括住，並以垂直線 (|) 隔開可能的選擇。當您指定這些屬性之一時，請不要在您的來源檔中包括大括弧，並指定顯示的選擇之一。

```
<data type="{ char | int | packed | zoned | float | byte | struct }"
    [ bidistringtype="{ ST4 | ST5 | ST6 | ST7 | ST8 | ST9 | ST10 | ST11 | DEFAULT }"]
    [ ccsid="{ number | data-name }" ]
    [ chartype="{ onebyte | twobyte }"]
    [ count="{ number | data-name }" ]
    [ init="string" ]
    [ length="{ number | data-name }" ]
    [ maxvrm="version-string" ]
    [ minvrm="version-string" ]
    [ name="name" ]
    [ offset="{ number | data-name }" ]
    [ offsetfrom="{ number | data-name | struct-name }" ]
    [ outputsize="{ number | data-name | struct-name }" ]
    [ passby= "{ reference | value }" ]
    [ precision="number" ]
    [ struct="struct-name" ]
    [ trim="{ right | left | both | none }" ]
    [ usage="{ inherit | input | output | inputoutput }" ]>
</data>
```

下表將列出資料標籤的屬性。每個項目都包括屬性名稱、可能的有效值和屬性的說明。

屬性	值	說明
type=	*char*，其中 *char* 表示字元值。*char* 資料值將以 *java.lang.String* 傳回。 詳細資訊，請參閱 *char* 長度值。 *int*，其中 *int* 爲一整數值。*int* 資料值將以 *java.lang.Long* 傳回。 詳細資訊，請參閱 *int* 長度和精準度的值。 *packed*，其中 *packed* 是壓縮十進位數值。*packed* 資料值將以 *java.math.BigDecimal* 傳回。詳細資訊，請參閱 *packed* 長度和精準度的值。 *zoned*，其中 *zoned* 是區化十進位值。*zoned* 資料值將以 *java.math.BigDecimal* 傳回。詳細資訊，請參閱 *zoned* 長度和精準度的值。 *float*，其中 *float* 是浮點值。**length** 屬性會指定位元組數 ″4″ 或 ″8″。4 位元組的整數將以 *java.lang.Float* 傳回。8 位元組整數以 *java.lang.Double* 傳回。詳細資訊，請參閱長度的 *float* 值。 *byte*，其中 *byte* 是位元組值。不對資料執行任何轉換。*byte* 資料值將以 *byte* 值 (*byte[]*) 的陣列傳回。 詳細資訊，請參閱長度的 *byte* 值。 *struct*，其中 *struct* 指定 **<struct>** 元素的名稱。*struct* 可讓您在定義結構一次之後，即可在文件內予以重複使用。當 **type=***″struct″* 時，它會像指定的結構一般出現在文件中的這個位置。*struct* 不使用長度值，也沒有精準度的值。	指出要使用的資料類型 (character, integer, packed, zoned, floating point, byte 或 struct)。 長度和精準度屬性的值會隨著資料類型而有所不同。詳細資訊，請參閱長度和精準度的值。

屬性	值	說明
bidistringtype=	*DEFAULT*，其中 DEFAULT 爲「非雙向資料 (LTR)」的預設字串類型。 *ST4*，其中 *ST4* 爲字串類型 4。 *ST5*，其中 *ST5* 爲字串類型 5。 *ST6*，其中 *ST6* 爲字串類型 6。 *ST7*， 其中 *ST7* 爲字串類型 7。 *ST8*，其中 *ST8* 爲字串類型 8。 *ST9*， 其中 *ST9* 爲字串類型 9。 *ST10*，其中 *ST10* 爲字串類型 10。 *ST11*， 其中 *ST11* 爲字串類型 11。	爲具有 **type=***"char"* 的 **<data>** 元素指定雙向字串類型。如果省略此屬性，此元素的字串類型將由 CCSID 所指定 (由其直接指定或爲主電腦環境的預設 CCSID)。 字串類型定義於 BidiStringType 類別的 javadoc 中。
ccsid=	*number*，其中 *number* 定義固定、永不變更的 CCSID。 *data-name*， 其中 *data-name* 定義在執行時間將含有字元資料的 CCSID 之名稱。指定的 *data-name* 可以是完整名稱，或是現行元素的相對名稱。不管是哪一種情況，名稱都必須參照以 **type=***"int"* 定義的 **<data>** 元素。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。	指定 **<data>** 元素的字元資料之主電腦「編碼字集 ID (CCSID)」。**ccsid** 屬性只能指定給具有 **type=***"char"* 的 **<data>** 元素。 如果略過這個屬性，這個元素的字元資料將假設具有主電腦環境的預設 CCSID。
chartype=	*onebyte*，其中 *onebyte* 指定每一個字元的大小。 *twobyte*， 其中 *twobyte* 指定每一個字元的大小。 使用 *chartype* 時，**length=***"number"* 屬性則是指定字元的數目，而非位元組的數目。	指定每一個字元的大小。
count=	*number*，其中 *number* 定義某種大小陣列中固定的、永不變更的元素數目。 *data-name*， 其中 *data-name* 會定義 PCML 文件中 **<data>** 元素的名稱，在執行時該名稱將含有陣列中的元素數目。指定的 *data-name* 可以是完整名稱，或是現行元素的相對名稱。不管是哪一種情況，名稱都必須參照以 **type=***"int"* 定義的 **<data>** 元素。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。	指定元素是一個陣列，且識別陣列中的登錄數目。 如果略過 *count* 屬性，則元素雖然可能包含在另一個定義爲陣列的元素內，但它還是不會定義爲陣列。

屬性	值	說明
init=	*string*	指定**<data>** 元素的起始值。當您以 **usage=***″input″* 或 **usage=***″inputoutput″* 來使用 **<data>** 元素時，如果應用程式沒有明確地設定起始值，便會使用 *init* 值。 指定的起始值係用來起始設定純量值。如果元素定義爲陣列或包含在定義爲陣列的結構內，則指定的起始值將作爲陣列中所有登錄的起始值。
length=	*number*，其中 *number* 定義資料所需的位元組數。不過，在使用 *chartype* 屬性時， *number* 指定的則是字元數，而非位元組數。 *data-name*，其中 *data-name* 會定義 PCML 文件中，將在執行時含有長度的 **<data>** 元素的名稱。*data-name* 僅能指定給 **type=***″char″* 或 **type=***″byte″* 的 **<data>** 元素。指定的 *data-name* 可以是完整名稱，或是現行元素的相對名稱。不管是哪一種情況，名稱都必須參照以 **type=***″int″* 定義的 **<data>** 元素。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。	指定資料元素的長度。這個屬性的用法會隨著資料類型而有所不同。詳細資訊，請參閱長度和精準度的值。
maxvrm=	*version-string*	指定此元素所在之處的最高 iSeries 版本。如果 iSeries 版本高於此屬性上指定的版本，則在呼叫程式時，將不會處理此元素及其子項 (如果有的話)。此屬性有助於定義 iSeries 版次間不同的程式介面。 版本字串的語法必須是 ″VvRrMm″， 其中大寫字母 ″V″、″R″ 及 ″M″ 是文字字元，而 ″v″、″r″ 及 ″m″ 則是代表版本、 版次及修正層次的數字 (一或多位數)。″v″ 的值必須是從 1 到 255。″r″ 及 ″m″ 的值必須是從 0 到 255。

屬性	值	說明
minvrm=	*version-string*	指定此元素所在之處的最低 iSeries 版本。如果 iSeries 版本低於此屬性上指定的版本，則在呼叫程式時，將不會處理此元素及其子項 (如果有的話)。此屬性有助於定義 iSeries 版次間不同的程式介面。 版本字串的語法必須是 ″VvRrMm″，其中大寫字母 ″V″、″R″ 及 ″M″ 是文字字元，而 ″v″、″r″ 及 ″m″ 則是代表版本、 版次及修正層次的數字 (一或多位數)。″v″ 的值必須是從 1 到 255。″r″ 及 ″m″ 的值必須是從 0 到 255。
name=	*name*	指定 **<data>** 元素的名稱。
offset=	*number*，其中 *number* 定義固定、永不變更的偏移。 *data-name*，其中 *data-name* 會定義 PCML 文件中，將在執行時含有此 **<data>** 元素的偏移之元素名稱。指定的 *data-name* 可以是完整名稱，或是現行元素的相對名稱。不管是哪一種情況，名稱都必須參照以 **type=**″*int*″ 定義的 **<data>** 元素。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。	指定輸出參數內的 **<data>** 元素的偏移。 有些程式會傳回具有固定結構的資訊，其後跟著一個或多個可變長度欄位或結構。在這種情況下，可變長度元素的位置，通常指定爲參數內的偏移或位移。 **offset** 屬性會與 **offsetfrom** 屬性一起使用。如果未指定 **offsetfrom** 屬性，則 **offset** 屬性上指定的偏移基本位置即爲這個元素的母項。請參閱指定偏移，以取得如何使用 **offset** 與 **offsetfrom** 屬性的詳細資訊。 **offset** 與 **offsetfrom** 屬性僅會用來處理來自程式的輸出資料。這些屬性不會控制輸入資料的位移或位置。 如果略過這個屬性，則這個元素的資料位置之後將緊跟著參數中的前一個元素，若有的話。

屬性	值	說明
offsetfrom=	*number*，其中 *number* 定義固定、永不變更的基本位置。*Number* 通常是最常用來指定 **number=***"0"*，表示位移是從參數開頭開始的絕對位移。 *data-name*，其中 *data-name* 會定義 **<data>** 元素的名稱，用來作爲偏移的基本位置。指定的元素名稱必須是這個元素的母項或祖先項。來自 **offset** 屬性的值，將相對於此屬性上指定元素的位置。指定的 *data-name* 可以是完整名稱，或是現行元素的相對名稱。不管哪一種情況，名稱必須參照這個元素的祖先項。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。 *struct-name*，其中 *struct-name* 會定義 **<struct>** 元素的名稱，用來作爲偏移的基本位置。指定的元素名稱必須是這個元素的母項或祖先項。來自 **offset** 屬性的值，將相對於此屬性上指定元素的位置。指定的 *struct-name* 可爲完整的名稱或是與現行元素相對的名稱。不管哪一種情況，名稱必須參照這個元素的祖先項。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。	指定與**位移**屬性相對的基本位置。 如果未指定 **offsetfrom** 屬性，則 **offset** 屬性上指定的偏移基本位置即爲這個元素的母項。請參閱指定偏移， 以取得如何使用 **offset** 與 **offsetfrom** 屬性的詳細資訊。 **offset** 與 **offsetfrom** 屬性僅會用來處理程式的輸出資料。這些屬性不會控制輸入資料的位移或位置。
outputsize=	*number*，其中 *number* 會定義要保留的固定、永不變更的位元組數目。 *data-name*， 其中 *data-name* 會定義 PCML 文件中 **<data>** 元素的名稱，在執行時該名稱將含有要爲輸出資料保留的位元組數。指定的 *data-name* 可以是完整名稱，或是現行元素的相對名稱。不管是哪一種情況，名稱都必須參照以 **type=***"int"* 定義的 **<data>** 元素。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。	指定要保留給元素的輸出資料的位元組數目。對長度可變的輸出參數而言，必須要有 **outputsize** 屬性，才能指定須保留多少個位元組讓從 iSeries 程式傳回的資料使用。**outputsize** 屬性可在所有可變長度欄位及可變大小陣列中指定，或可針對含有一或多個可變長度欄位的整個參數指定。 沒有必要針對固定大小輸出參數指定 **Outputsize**，就算有必要也不得指定。 這個屬性上指定的值將作爲元素的總大小，包括元素的所有子項。因此，在元素的子項或後代項上，不會處理 **outputsize** 屬性。 如果省略 **outputsize**，則要保留給輸出資料的位元組數，將在爲 **<struct>** 元素的所有子項新增要保留的位元組數時決定。

屬性	值	說明
passby=	*reference*，其中 *reference* 表示參數將由參照來傳送。當程式被呼叫時，指標會傳送參數值到程式。 *value*，其中 *value* 表示一整數值。只有在指定了 **type=** *"int"* 以及 **length=***"4"* 時，才能使用此值。	指定參數是否由參照傳送或由值傳送。只有在此元素爲定義服務程式呼叫的 **<program>** 元素子項時，才能使用此屬性。
precision=	*number*	指定某些數值資料類型的精確度的位元組數目。詳細資訊，請參閱長度和精準度的值。
struct=	*name*	指定 **<data>** 元素的 **<struct>** 元素名稱。 **struct** 屬性僅能指定給其 **type=***"struct"* 的 **<data>** 元素。
trim=	*right*，其中 *right* 爲表示會刪除尾端白色空間的預設行爲。 *left*，其中 *left* 表示將刪除前導的白色空間。 *both*，其中 *both* 表示前導及尾端白色空間都要刪除。 *none*， 其中 *none* 表示不會刪除白色空間。	指定如何刪除字元資料中的白色空間。
usage=	*inherit*	用法將繼承自母項元素。如果結構沒有母項，則用法將假設爲 *inputoutput*。
	input	定義主程式的輸入值。對於字元與數字類型，將執行適當的轉換。
	output	定義主程式的輸出值。對於字元與數字類型，將執行適當的轉換。
	inputoutput	定義輸入及輸出值。

指定位移

有些程式會傳回具有固定結構的資訊，其後跟著一個或多個可變長度欄位或結構。在這種情況下，可變長度元素的位置，通常指定爲參數內的偏移或位移。

位移即是從參數開頭到欄位或結構開頭的距離，以位元組表示。位置即是從某個結構的開頭到另一結構開頭的距離，以位元組表示。

對於偏移，因爲距離是從參數開頭開始算起，所以您必須指定 **offsetfrom=″0″**。下列是從參數開頭開始算起的位移的範例：

```
<pcml version="1.0">
  <program name="myprog" path="/QSYS.lib/MYLIB.lib/MYPROG.pgm">
    <!-- receiver variable contains a path -->
    <struct name="receiver" usage="output" outputsize="2048">
      <data name="pathType"         type="int"  length="4" />
      <data name="offsetToPathName" type="int"  length="4" />
      <data name="lengthOfPathName" type="int"  length="4" />
      <data name="pathName"         type="char" length="lengthOfPathName"
```

```
              offset="offsetToPathName"  offsetfrom="0"/>
    </struct>
  </program>
</pcml>
```

對於位置，因爲距離是從另一個結構開始算起，所以您可以指定與位移相對的結構的名稱。下列是從已命名的結構開頭開始算起的位置的範例：

```
<pcml version="1.0">
  <program name="myprog" path="/QSYS.lib/MYLIB.lib/MYPROG.pgm">
    <!-- receiver variable contains an object -->
    <struct name="receiver" usage="output" >
      <data name="objectName"       type="char"  length="10" />
      <data name="libraryName"      type="char"  length="10" />
      <data name="objectType"       type="char"  length="10" />
      <struct name="pathInfo" usage="output" outputsize="2048" >
        <data name="pathType"         type="int"  length="4" />
        <data name="offsetToPathName" type="int"  length="4" />
        <data name="lengthOfPathName" type="int"  length="4" />
        <data name="pathName"         type="char" length="lengthOfPathName"
                offset="offsetToPathName"  offsetfrom="pathInfo"/>
      </struct>
    </struct>
  </program>
</pcml>
```

長度與精準度的值:

長度和精準度屬性的值會隨著資料類型而有所不同。

下表列出每一個資料類型的長度與精準度可能值說明。

資料類型	長度	精準度
type="char"	此元素資料的位元組數，不一定是字元數。您必須指定文字*長度* 或*資料名稱*。	未提供
type="int"	此元素資料的位元組數：2、4 或 8。您必須指定文字*長度*。	指出精準度的位元數，及整數是否帶正負號或無正負號： • 就 **length**="2" 而言 – 使用 **precision**="15" 代表帶正負號的 2 位元組整數。 此爲預設值 – 使用 **precision**="16" 代表無正負號的 2 位元組整數 • 就 **length**="4" 而言 – 使用 **precision**="31" 代表帶正負號的 4 位元組整數 – 使用 **precision**="32" 代表無正負號的 4 位元組整數 • 就 **length**="8" 而言，使用 **precision**="63" 代表帶正負號的 8 位元組整數
type="packed" 或 "zoned"	此元素資料之位數。您必須指定文字*長度*。	元素的十進位數。此數字必須大於或等於 0，且小於或等於 **length** 屬性中指定的總位數。
type="float"	此元素資料之位元組數，4 或 8。您必須指定文字*長度*。	未提供

資料類型	長度	精準度
type="byte"	此元素資料之位元組數。您必須指定文字*長度*或*資料名稱*。	未提供
type="struct"	不允許。	未提供

解析相對名稱

有幾個屬性可讓您指定文件內另一個元素或標籤的名稱，作爲屬性值。指定的名稱可以是與現行標籤相對的名稱。

可以經由查看名稱可否解析爲含有現行標籤的標籤的子項或後代項，來解析名稱。如果無法在這個層次解析名稱，則將在下一含有標籤的最高層次中繼續搜尋。此解析最後必須產生與 **<pcml>** 或 **<rfml>** 標籤所包含完全一樣的標籤，在此情況下，名稱會被視爲絕對名稱而非相對名稱。

以下爲使用 PCML 的範例：

```
<pcml version="1.0">
  <program name="polytest" path="/QSYS.lib/MYLIB.lib/POLYTEST.pgm">
    <!-- Parameter 1 contains a count of polygons along with an array of polygons -->
    <struct name="parm1" usage="inputoutput">
      <data name="nbrPolygons" type="int" length="4" init="5" />
      <!-- Each polygon contains a count of the number of points along with an array of points -->
      <struct name="polygon" count="nbrPolygons">
        <data name="nbrPoints" type="int" length="4" init="3" />
        <struct name="point" count="nbrPoints" >
          <data name="x" type="int" length="4" init="100" />
          <data name="y" type="int" length="4" init="200" />
        </struct>
      </struct>
    </struct>
  </program>
</pcml>
```

以下是使用 RFML 的範例：

```
<rfml version="4.0">
  <struct name="polygon">
    <!-- Each polygon contains a count of the number of points along with an array of points. -->
    <data name="nbrPoints" type="int" length="4" init="3" />
    <data name="point" type="struct" struct="point" count="nbrPoints" />
  </struct>
  <struct name="point" >
    <data name="x" type="int" length="4" init="100" />
    <data name="y" type="int" length="4" init="200" />
  </struct>
  <recordformat name="polytest">
    <!-- This format contains a count of polygons along with an array of polygons -->
    <data name="nbrPolygons" type="int" length="4" init="5" />
    <data name="polygon" type="struct" struct="polygon" count="nbrPolygons" />
  </recordformat>
</rfml>
```

記錄格式標記語言

記錄格式標記語言 (RFML) 是指定記錄格式用的 XML 延伸語言。

IBM Toolbox for Java RFML 元件可讓您的 Java 應用程式使用 RFML 文件來指定及操作某些種類記錄內的欄位。

RFML 文件稱爲 RFML 來源檔，代表針對 iSeries 伺服器上的實體及邏輯檔案所定義之資料說明規格 (DDS) 資料類型的實用子集。您可以使用 RFML 文件來管理下列當中的資訊：

- 檔案記錄
- 資料佇列登錄
- 使用者空間
- 自訂資料緩衝區

註: 有關使用 DDS 來說明資料屬性的詳細資訊，請參閱 DDS 參照。

RFML 非常類似程式呼叫標記語言 (PCML)，PCML 是 IBM Toolbox for Java 所支援的另一個 XML 延伸規格。RFML 不是 PCML 的子集也不是 PCML 的超集，而是新增幾個新的元素和屬性，並省略不執行一些元素和屬性的一種兄弟語言。

PCML 可作爲使用 ProgramCall 和 ProgramParameter 類別的 XML 導向的選擇方案。 同樣地，RFML 也是 Record、RecordFormat 和 FieldDescription 等類別另一個具有使用親和力、又容易維護的選擇方案。

若需 RFML 的其它相關資訊，請參閱下列主題：

基本要求
閱讀有關使用 RFML 的基本要求。

RFML 範例
說明在您的應用程式中使用 RFML 時，將會如何減少、甚至簡化您必須撰寫的程式碼。本範例中有 RFML 來源檔的範例。

RecordFormatDocument 類別
閱讀如何與其他 Toolbox for Java 類別一起使用 RecordFormatDocument 類別，以讀取及寫入資料。

RFML 文件與 RFML 語法
瞭解 RFML 資料類型定義中所定義的 RFML 文件 (稱爲 RFML 來源檔) 以及 RFML 語法。

RFML 只是把 XML 使用在您的伺服器上的一種方式。如需在 iSeries 伺服器中使用 XML 的相關資訊，請參閱 IBM Toolbox for Java XML 延伸規格，以及可延伸標記語言 (XML)。

使用 RFML 的基本要求

RFML 元件的工作站 Java 虛擬機器基本要求與其他 IBM Toolbox for Java 相同。

此外，爲了要在執行時間剖析 RFML，應用程式的 CLASSPATH 必須包含 XML 剖析器。XML 剖析器必須延伸 org.apache.xerces.parsers.SAXParser 類別。如需詳細資訊，請參閱下列網頁：

第 370 頁的『XML 剖析器與 XSLT 處理器』

註: RFML 的剖析器基本要求與 PCML 的相同。與 PCML 一樣，如果您預先序列化 RFML 檔案， 則不必把 XML 剖析器併入應用程式的 CLASSPATH 即可執行應用程式。

相關參考

第 8 頁的『IBM Toolbox for Java 的工作站基本要求』
請確定您的工作站符合下列基本要求。

範例：使用 RFML 與使用 IBM Toolbox for Java Record 類別兩者的比較

本範例說明使用 RFML 與使用 IBM Toolbox for Java Record 類別兩者有何不同。

使用傳統式的 Record 類別，資料格式規格會與您的應用程式的商務邏輯交織在一起。新增、變更或刪除欄位表示您必須編輯並重新編譯 Java 程式碼。不過， 使用 RFML 可使資料格式規格隔離在 RFML 來源檔中，與商務邏輯完全區隔開來。容許欄位變更意味著 RFML 檔案的修改，通常不必變更或重新編譯 Java 應用程式。

本範例假設您的應用程式負責處理客戶記錄，您已將記錄定義於 RFML 來源檔，命名爲 qcustcdt.rfml。來源檔代表構成各筆客戶記錄的欄位。

以下報表說明 Java 應用程式將如何使用 IBM Toolbox for Java Record、RecordFormat 及 FieldDescription 類別，來解譯客戶記錄：

```
// Buffer containing the binary representation of one record of information.
    byte[] bytes;

    // ... Read the record data into the buffer ...

    // Set up a RecordFormat object to represent one customer record.
    RecordFormat recFmt1 = new RecordFormat("cusrec");
    recFmt1.addFieldDescription(new ZonedDecimalFieldDescription(new AS400ZonedDecimal(6, 0), "cusnum"));
    recFmt1.addFieldDescription(new CharacterFieldDescription(new AS400Text(8, 37), "lstnam"));
    recFmt1.addFieldDescription(new CharacterFieldDescription(new AS400Text(3, 37), "init"));
    recFmt1.addFieldDescription(new CharacterFieldDescription(new AS400Text(13, 37), "street"));
    recFmt1.addFieldDescription(new CharacterFieldDescription(new AS400Text(6, 37), "city"));
    recFmt1.addFieldDescription(new CharacterFieldDescription(new AS400Text(2, 37), "state"));
    recFmt1.addFieldDescription(new ZonedDecimalFieldDescription(new AS400ZonedDecimal(5, 0), "zipcod"));
    recFmt1.addFieldDescription(new ZonedDecimalFieldDescription(new AS400ZonedDecimal(4, 0), "cdtlmt"));
    recFmt1.addFieldDescription(new ZonedDecimalFieldDescription(new AS400ZonedDecimal(1, 0), "chgcod"));
    recFmt1.addFieldDescription(new ZonedDecimalFieldDescription(new AS400ZonedDecimal(6, 2), "baldue"));
    recFmt1.addFieldDescription(new ZonedDecimalFieldDescription(new AS400ZonedDecimal(6, 2), "cdtdue"));

    // Read the byte buffer into the RecordFormatDocument object.
    Record rec1 = new Record(recFmt1, bytes);

    // Get the field values.
    System.out.println("cusnum: " + rec1.getField("cusnum"));
    System.out.println("lstnam: " + rec1.getField("lstnam"));
    System.out.println("init:   " + rec1.getField("init"));
    System.out.println("street: " + rec1.getField("street"));
    System.out.println("city:   " + rec1.getField("city"));
    System.out.println("state:  " + rec1.getField("state"));
    System.out.println("zipcod: " + rec1.getField("zipcod"));
    System.out.println("cdtlmt: " + rec1.getField("cdtlmt"));
    System.out.println("chgcod: " + rec1.getField("chgcod"));
    System.out.println("baldue: " + rec1.getField("baldue"));
    System.out.println("cdtdue: " + rec1.getField("cdtdue"));
```

相比之下，改用 RFML 解譯相同記錄的情形如下。

使用 RFML 來解譯客戶資料記錄內容的 Java 程式碼可能如下：

```
// Buffer containing the binary representation of one record of information.
    byte[] bytes;

// ... Read the record data into the buffer ...

// Parse the RFML file into a RecordFormatDocument object.
// The RFML source file is called qcustcdt.rfml.
RecordFormatDocument rfml1 = new RecordFormatDocument("qcustcdt");

// Read the byte buffer into the RecordFormatDocument object.
rfml1.setValues("cusrec", bytes);
```

```
// Get the field values.
System.out.println("cusnum: " + rfml1.getValue("cusrec.cusnum"));
System.out.println("lstnam: " + rfml1.getValue("cusrec.lstnam"));
System.out.println("init:   " + rfml1.getValue("cusrec.init"));
System.out.println("street: " + rfml1.getValue("cusrec.street"));
System.out.println("city:   " + rfml1.getValue("cusrec.city"));
System.out.println("state:  " + rfml1.getValue("cusrec.state"));
System.out.println("zipcod: " + rfml1.getValue("cusrec.zipcod"));
System.out.println("cdtlmt: " + rfml1.getValue("cusrec.cdtlmt"));
System.out.println("chgcod: " + rfml1.getValue("cusrec.chgcod"));
System.out.println("baldue: " + rfml1.getValue("cusrec.baldue"));
System.out.println("cdtdue: " + rfml1.getValue("cusrec.cdtdue"));
```

RecordFormatDocument 類別

RecordFormatDocument 類別可讓您的 Java 程式在 RFML 的資料表示與 Record 及 RecordFormat 物件之間轉換，以便與其他 IBM Toolbox for Java 元件搭配使用。

RecordFormatDocument 類別

RecordFormatDocument 類別代表 RFML 來源檔，它所提供的方法可讓您的 Java 程式執行下列動作：

- 從 Record 物件、RecordFormat 物件和位元組陣列撰寫 RFML 來源檔
- 建立 Record 物件、RecordFormat 物件和位元組陣列撰寫 RFML 來源檔，表示出 RecordFormatDocument 物件所包含的資訊
- 取得及設定不同物件和資料類型的值
- 建立 XML (RFML)，代表 RecordFormatDocument 物件所包含的資料
- 序列化 RecordFormatDocument 物件代表的 RFML 來源檔

有關可用的方法之相關資訊，請參閱 javadoc 方法摘要中的 RecordFormatDocument 類別章節。

搭配其他 IBM Toolbox for Java 類別來使用 RecordFormatDocument 類別

請搭配下列 IBM Toolbox for Java 類別來使用 RecordFormatDocument 類別：

- 記錄導向的類別，包括讀取、操作和撰寫 Record 物件的記錄層級存取檔案類別 (AS400File、SequentialFile 和 KeyedFile)。此種類還包括 LineDataRecordWriter 類別。
- 位元組導向的類別，包括一次讀取和寫入資料的位元組陣列的某些 DataQueue、UserSpace 和 IFSFile 類別。

使用 RecordFormatDocument 類別時，請勿搭配下列 IBM Toolbox for Java 類別，RecordFormatDocument 將無法處理這些類別讀取及寫入資料所使用的格式：

- DataArea 類別，因爲它的讀取和寫入方法只處理「字串」、布林和 BigDecimal 等資料類型。
- IFSTextFileInputStream 和 IFSTextFileOutputStream，因爲這些讀取和寫入方法只處理「字串」。
- JDBC 類別，因爲 RFML 只處理 iSeries 資料說明規格 (DDS) 所說明的資料。

記錄格式文件與 RFML 語法

RFML 文件稱爲 RFML 來源檔，包含爲特定資料格式定義規格用的標籤。

因爲 RFML 是基於 PCML，所以它的語法爲 PCML 使用者所熟悉。因 爲 RFML 是 XML 的延伸，所以 RFML 來源檔既容易讀取，也很容易建立。例如，使用簡式的文字編輯器就可以建立 RFML 來源檔。同時，RFML 來源檔顯示資料結構的方式也比 Java 這些程式設計語言更容易瞭解。

RFML 範例使用 RFML 與使用 IBM Toolbox for Java Record 類別兩者的比較中，含有 RFML 來源檔這個範例。

RFML DTD

RFML 文件類型定義 (DTD) 定義出有效的 RFML 元素和語法。 爲確定 XML 剖析器可以在執行時間驗證您的 RFML 來源檔，請在原始檔中宣告 RFML DTD：

```
<!DOCTYPE rfml SYSTEM "rfml.dtd">
```

RFML DTD 常駐於 jt400.jar 檔案 (com/ibm/as400/data/rfml.dtd)。

RFML 語法

RFML DTD 定義出標籤，每一個都有自己的屬性標籤。RFML 標籤用來宣告和定義 RFML 來源檔中的下列元素：

- rfml 標籤開始和結束說明資料格式的 RFML 來源檔。
- struct 標籤定義出具名的結構， 您可以在 RFML 來源檔中重覆使用。結構中含有結構中每一個欄位的資料標籤。
- recordformat 標籤定義出記錄格式，當中不是含有資料元素，就是含有結構元素的參照。
- data 標籤定義出記錄格式或結構內的欄位。

以下範例中，RFML 語法說明一個記錄格式和一個結構：

```
<rfml>

    <recordformat>
        <data> </data>
    </recordformat>

    <struct>
        <data> </data>
    </struct>

</rfml>
```

RFML 文件類型定義 (DTD):

這是 RFML DTD。注意版本是 4.0。RFML DTD 常駐於 jt400.jar 檔案 (com/ibm/as400/data/rfml.dtd)。

```
<!--
Record Format Markup Language (RFML) Document Type Definition.

RFML is an XML language.  Typical usage:
   <?xml version="1.0"?>
   <!DOCTYPE rfml SYSTEM "rfml.dtd">
   <rfml version="4.0">
   ...
   </rfml>


(C) Copyright IBM Corporation, 2001,2002
All rights reserved.
Licensed Materials Property of IBM
US Government Users Restricted Rights
Use, duplication or disclosure restricted by
```

```
GSA ADP Schedule Contract with IBM Corp.
-->

<!-- Convenience entities -->
<!ENTITY % string             "CDATA">    <!-- a string of length 0 or greater -->
<!ENTITY % nonNegativeInteger "CDATA">    <!-- a non-negative integer -->
<!ENTITY % binary2            "CDATA">    <!-- an integer in range 0-65535 -->
<!ENTITY % boolean            "(true|false)">
<!ENTITY % datatype  "(char | int | packed | zoned | float | byte | struct)">
<!ENTITY % biditype  "(ST4 | ST5 | ST6 | ST7 | ST8 | ST9 | ST10 | ST11 | DEFAULT)">


<!-- The document root element -->
<!ELEMENT rfml (struct | recordformat)+>
<!ATTLIST rfml
          version       %string;    #FIXED "4.0"
                    ccsid         %binary2;             #IMPLIED
>
<!-- Note: The ccsid is the default value that will be used for -->
     <!-- any contained <data type="char"> elements that do not specify a ccsid. -->


<!-- Note: RFML does not support nested struct declarations.  -->
<!-- All struct elements are direct children of the root node. -->
<!ELEMENT struct (data)+>
<!ATTLIST struct
          name          ID          #REQUIRED
>


<!-- <!ELEMENT recordformat (data | struct)*> -->
<!ELEMENT recordformat (data)*>
<!ATTLIST recordformat
          name          ID                    #REQUIRED
          description   %string;              #IMPLIED
>
<!-- Note: On the server, the Record "text description" field is limited to 50 bytes. -->

<!ELEMENT data EMPTY>
<!ATTLIST data
          name          %string;              #REQUIRED

          count         %nonNegativeInteger;  #IMPLIED

          type          %datatype;            #REQUIRED
          length        %nonNegativeInteger;  #IMPLIED
          precision     %nonNegativeInteger;  #IMPLIED
          ccsid         %binary2;             #IMPLIED
          init          CDATA                 #IMPLIED
          struct        IDREF                 #IMPLIED

          bidistringtype %biditype;           #IMPLIED
>
<!-- Note: The 'name' attribute must be unique within a given recordformat. -->
<!-- Note: On the server, the length of Record field names is limited to 10 bytes. -->
<!-- Note: The 'length' attribute is required, except when type="struct". -->
<!-- Note: If type="struct", then the 'struct' attribute is required. -->
<!-- Note: The 'ccsid' and 'bidistringtype' attributes are valid only when type="char". -->
<!-- Note: The 'precision' attribute is valid only for types "int", "packed", and "zoned". -->


<!-- The standard predefined character entities -->
<!ENTITY quot  "&#34;">     <!-- quotation mark -->
<!ENTITY amp   "&#38;#38;"> <!-- ampersand -->
<!ENTITY apos  "&#39;">     <!-- apostrophe -->
<!ENTITY lt    "&#38;#60;"> <!-- less than -->
<!ENTITY gt    "&#62;">     <!-- greater than -->
```

```
<!ENTITY nbsp  " ">    <!-- non-breaking space -->
<!ENTITY shy   "&#173;">    <!-- soft hyphen (discretionary hyphen) -->
<!ENTITY mdash "&#38;#x2014;">
<!ENTITY ldquo "&#38;#x201C;">
<!ENTITY rdquo "&#38;#x201D;">
```

RFML 資料標籤:

資料標籤具有下列屬性。

若以方括弧 ([]) 括住，即代表選用性的屬性。如果您指定了選用性屬性，請不要在您的原始程式中使用方括弧 ([])。有些屬性值會以選擇清單 (以大括弧括住) 顯示，並以垂直線區隔可能的選擇。當您指定這些屬性之一時，請不要在您的來源檔中包括大括弧，並指定顯示的選擇之一。

```
<data type="{ char | int | packed | zoned | float | byte | struct }" ]
    [ bidistringtype="{ ST4 | ST5 | ST6 | ST7 | ST8 | ST9 | ST10 | ST11 | DEFAULT }"]
    [ ccsid="{ number | data-name }" ]
    [ count="{ number | data-name }" ]
    [ init="string" ]
    [ length="{ number | data-name }" ]
    [ name="name" ]
    [ precision="number" ]
    [ struct="struct-name" ]>
</data>
```

下表將列出資料標籤的屬性。每個登錄都包括屬性名稱、可能的有效值和屬性的說明。

屬性	值	說明
type=	*char* 爲字元值。*char* 資料值將以 *java.lang.String* 傳回。 詳細資訊，請參閱 *char* 長度值。 *int* 爲整數值。*int* 資料值將以 *java.lang.Long* 傳回。 詳細資訊，請參閱 *int* 長度和精準度的值。 *packed* 爲壓縮的十進位值。 *packed* 資料值將以 *java.math.BigDecimal* 傳回。詳細資訊，請參閱 *packed* 長度和精準度的值。 *zoned* 爲區化的十進位值。 *zoned* 資料值將以 *java.math.BigDecimal* 傳回。詳細資訊，請參閱 *zoned* 長度和精準度的值。 *float* 爲浮點值。 **length** 屬性指定位元組數目爲：4 或 8。 4 位元組的整數將以 *java.lang.Float* 傳回。8 位元組整數以 *java.lang.Double* 傳回。詳細資訊，請參閱長度的 *float* 值。 *byte* 爲位元組值。不對資料執行任何轉換。*byte* 資料值將以 *byte* 值 (*byte[]*) 的陣列傳回。 詳細資訊，請參閱長度的 *byte* 值。 *struct* 爲 **<struct>** 元素的名稱。*struct* 可讓您定義結構一次，並在文件內重複使用它多次。當您使用 **type=**″struct″ 時，它就會像指定的結構一般出現在文件中的這個位置上。*struct* 不使用長度值，也沒有精準度的值。	指出要使用的資料類型 (character, integer, packed, zoned, floating point, byte 或 struct)。 長度和精準度屬性的值會隨著資料類型而有所不同。詳細資訊，請參閱長度和精準度的值。
bidistringtype=	*DEFAULT*，其中 DEFAULT 爲「非雙向資料 (LTR)」的預設字串類型。 *ST4*，其中 *ST4* 爲字串類型 4。 *ST5*，其中 *ST5* 爲字串類型 5。 *ST6*，其中 *ST6* 爲字串類型 6。 *ST7*， 其中 *ST7* 爲字串類型 7。 *ST8*，其中 *ST8* 爲字串類型 8。 *ST9*， 其中 *ST9* 爲字串類型 9。 *ST10*，其中 *ST10* 爲字串類型 10。 *ST11*， 其中 *ST11* 爲字串類型 11。	針對具有 **type=**″char″ 的 **<data>** 元素指定雙向字串類型。如果略過這個屬性，此元素的字串類型將由 CCSID 所隱含 (明確指定或主電腦環境的預設 CCSID)。 字串類型定義於 BidiStringType 類別的 javadoc 中。

屬性	值	說明
ccsid=	*number*，其中 *number* 定義固定、永不變更的 CCSID。 *data-name*， 其中 *data-name* 定義在執行時間將含有字元資料的 CCSID 之名稱。指定的 *data-name* 可以是完整名稱，或是現行元素的相對名稱。不管是哪一種情況，名稱都必須參照以 **type="int"** 定義的 **<data>** 元素。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。	爲 **<data>** 元素指定字元資料的主電腦編碼字集 ID (CCSID)。**ccsid** 屬性只能指定給具有 **type="char"** 的 **<data>** 元素。 如果略過這個屬性，這個元素的字元資料將假設具有主電腦環境的預設 CCSID。
count=	*number*，其中 *number* 定義某種大小陣列中固定的、永不變更的元素數目。 *data-name*， 其中 *data-name* 會定義 RFML 文件中 **<data>** 元素的名稱，在執行時該名稱將含有陣列中的元素數目。指定的 *data-name* 可以是完整名稱，或是現行元素的相對名稱。不管是哪一種情況，名稱都必須參照以 **type="int"** 定義的 **<data>** 元素。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。	指定元素是一個陣列，且識別陣列中的登錄數目。 如果略過 count 屬性，則元素雖然可能包含在另一個定義爲陣列的元素內，但它還是不會定義爲陣列。
init=	*string*	指定**<data>** 元素的起始值。 指定的起始值係用來起始設定純量值。如果元素定義爲陣列或包含在定義爲陣列的結構內，則指定的起始值將作爲陣列中所有登錄的起始值。
length=	*number*，其中 *number* 定義固定、永不變更的長度。 *data-name*，其中 *data-name* 會定義 RFML 文件中，將在執行時含有長度的 **<data>** 元素的名稱。*data-name* 僅能指定給 **type="char"** 或 **type="byte"** 的 **<data>** 元素。指定的 *data-name* 可以是完整名稱，或是現行元素的相對名稱。不管是哪一種情況，名稱都必須參照以 **type="int"** 定義的 **<data>** 元素。請參閱解析相對名稱，以取得如何解析相對名稱的詳細資訊。	指定資料元素的長度。這個屬性的用法會隨著資料類型而有所不同。詳細資訊，請參閱長度和精準度的值。
name=	*name*	指定 **<data>** 元素的名稱。
precision=	*number*	指定某些數值資料類型的精確度的位元組數目。詳細資訊，請參閱長度和精準度的值。

屬性	值	說明
struct=	*name*	指定 **<data>** 元素的 **<struct>** 元素名稱。**struct** 屬性僅能指定給 **type=**″struct″ 的 **<data>** 元素。

RFML rfml 標籤:

rfml 標籤可具有下列屬性；

若以方括弧 ([]) 括住，即代表選用性的屬性。如果您指定一個選用性屬性，請不要在您的來源檔中包括括號 ([])。

```
<rfml version="version-string"
    [ ccsid="number" ]>
</rfml>
```

以下表格列出 rfml 標籤的屬性。每個登錄都包括屬性名稱、可能的有效值和屬性的說明。

屬性	值	說明
version=	*version-string* 爲固定版本的 RFML DTD。就 V5R3 而言， 4.0 是唯一的有效值。	指定 RFML DTD 的版本，您可以用來驗證值是否正確。
ccsid=	*number* 爲固定、永不變更的編碼字集 ID (CCSID)。	指定主電腦 CCSID，套用到所有含括在內而未指定 CCSID 的 **<data type=**″char″> 元素。如需詳細資訊，請參閱 RFML **<data>** 標籤。當您省略不執行這個屬性時，則使用主電腦環境的預設 CCSID。

RFML recordformat 標籤:

recordformat 標籤可具有下列屬性。

若以方括弧 ([]) 括住，即代表選用性的屬性。如果您指定一個選用性屬性，請不要在您的來源檔中包括括號 ([])。

```
<recordformat name="name"
    [ description="description" ]>
</recordformat>
```

以下表格列出 recordformat 標籤的屬性。每個登錄都包括屬性名稱、可能的有效值和屬性的說明。

屬性	值	說明
name=	*name*	指定 recordformat 的名稱。
description=	*說明*	指定 recordformat 的說明。

RFML struct 標籤:

struct 標籤可具有下列屬性；

若以方括弧 ([]) 括住，即代表選用性的屬性。如果您指定一個選用性屬性，請不要在您的來源檔中包括括號 ([])。

```
<struct name="name">
</struct>
```

以下表格列出 struct 標籤的屬性。每個登錄都包括屬性名稱、可能的有效值和屬性的說明。

屬性	值	說明
name=	*name*	指定 **<struct>** 元素的名稱。

XML 剖析器與 XSLT 處理器

某些 IBM Toolbox for Java 套件或功能會要求，在執行時間，您的 CLASSPATH 環境變數中必須具備「可延伸標記語言 (XML)」剖析器或「可延伸樣式表語言轉換 (XSLT)」處理器。

請參閱下列資訊，決定您要使用的剖析器或處理器。

如需哪些 IBM Toolbox for Java 套件及功能需要 XML 剖析器或 XSLT 處理器的相關資訊，請參閱下列頁面：

第 10 頁的『Jar 檔』

XML 剖析器

如果套件需要 XML 剖析器，您就必須於執行時間在 CLASSPATH 環境變數中予以納入。該 XML Parser 必須符合下列要求：

- 符合 JAXP
- 延伸類別 org.apache.xerces.parsers.SAXParser
- 支援完整綱目驗證

註: 如果要使用 XPCML，則剖析器只要支援完整綱目驗證即可。如果您只要使用 PCML， 則不必進行完整綱目驗證。

Java 2 Software Developer Kit (J2SDK) 1.4 版包含了適合的 XML 剖析器。如果您使用的是舊版的 J2SDK，請使用下列任一方法來存取適合的 XML 剖析器：

- 使用 x4j400.jar (由 Apache 所提供的 IBM 版 Xerces XML 剖析器)
- 從 Apache Web site  下載 XercesXML Parser。
- 使用 iSeries 伺服器上 /QIBM/ProdData/OS400/xml/lib 目錄中任何相容的 XML 剖析器

註: 考慮使用 /QIBM/ProdData/OS400/xml/lib 中最新版本的剖析器。例如，xmlapis11.jar 和 xerces411.jar 兩個都是完整驗證的剖析器。

您可以在伺服器上使用這些剖析器，或將它們複製到工作站。

註: 任何符合執行 XPCML 要求的 XML 剖析器，都能執行 PCML 和 RFML。請記得，XPCML 並不支援序列化。

XSLT 處理器

如果套件需要 XSLT 處理器，您就必須於執行時間在 CLASSPATH 環境變數中予以納入。該 XSLT 處理器必須符合下列要求：

- 符合 JAXP
- 包含類別 javax.xml.transform.Transformer

Java 2 Software Developer Kit (J2SDK) 1.4 版包含了適合的 XSLT 處理器。如果您使用的是舊版的 J2SDK，請使用下列任一方法來存取適合的 XSLT 處理器：

- 使用 xslparser.jar (由 Apache 所提供的 IBM 版 Xalan XSLT 處理器)
- 從 Apache Web site 下載 Xalan XSLT 處理器。
- 使用 iSeries 伺服器上 /QIBM/ProdData/OS400/xml/lib 目錄中任何相容的 XSLT 處理器

您可以在伺服器上使用這些處理器，或將它們複製到工作站。

可延伸的程式呼叫標記語言

「可延伸程式呼叫標記語言 (XPCML)」可提供對 XML 綱目的支援， 進而加強「程式呼叫標記語言 (PCML)」的功能性及實用性。XPCML 不像 PCML 那樣能支援序列化，因此， 您無法序列化 XPCML 文件。

然而相較於 PCML，XPCML 提供了多項有益的加強功能：

- 指定與傳送程式參數值
- 以 XPCML 來擷取對 iSeries 伺服器的程式呼叫結果
- 將現有的 PCML 文件轉換爲同等的 XPCML 文件
- 延伸與自訂 XPCML 綱目，以定義新的簡式或複合元素與屬性

有關 XPCML 的詳細資訊，請參閱下列頁面：

XPCML 相對於 PCML 的優勢
瞭解 XPCML 相對於 PCML 的優勢相關資訊。

基本要求
閱讀使用 XPCML 的軟體需求相關資訊。

XPCML 綱目及語法
瞭解 XPCML 綱目定義 XPCML 語法的方式以及可用的伺服器資料類型。瞭解如何延伸與自訂綱目，以符合特定的程式設計需要。

使用 XPCML
瞭解如何利用 XPCML 加強功能的優勢，包括傳送不同類型的參數到伺服器程式， 以及擷取傳回的參數資料。瞭解如何壓縮 XPCML 程式碼，讓程式碼更容易使用與閱讀，並且瞭解如何搭配 XPCML 使用您目前具有 PCML 功能的應用程式。

XPCML 就是一種在伺服器上使用 XML 的方式。有關使用 XML 的詳細資訊，請參閱下列頁面：

可延伸標記語言元件

XML 工具程式

W3C Architecture domain: XML Schema

XPCML 相對於 PCML 的優勢

「可延伸程式呼叫標記語言 (XPCML)」對 PCML 提供了更多有用的加強功能。

- 指定與傳送程式參數值
- 以 XPCML 來擷取對 iSeries 伺服器的程式呼叫結果
- 將現有的 PCML 文件轉換爲同等的 XPCML 文件
- 延伸與自訂 XPCML 綱目，以定義新的簡式或複合元素與屬性

指定與傳送程式參數值

XPCML 會使用 XML 綱目來定義程式參數型類；PCML 會使用資料類型定義 (DTD)。在剖析時，XML Parser 會依據綱目中所定義的適當參數來驗證輸入爲參數的資料值。作爲參數的資料類型有很多種：字串、整數以及 Long 等等。指定與傳送程式參數值這項功能，是相對於 PCML 的一項重大改進。在 PCML 中，您只能在剖析 PCML 文件之後才能驗證參數。此外，您通常必須編碼應用程式，才能在 PCML 中執行參數值的驗證。

以 XPCML 來擷取程式呼叫結果

XPCML 還提供這項功能， 讓您擷取程式呼叫結果作爲 XPCML。在 PCML 中，當您對程式進行呼叫之後，您必須呼叫 ProgramCallDocument 類別的其中一個 getValue 方法，才能取得程式呼叫結果。在 XPCML 中，您可以使用 getValue 方法， 但也可以讓 XPCML 呼叫 generateXPCML 方法，以取得作爲 XPCML 的程式呼叫結果。

將現有的 PCML 文件轉換為 XPCML

ProgramCallDocument 類別的一個新方法 transformPCMLToXPCML，可讓您將現有的 PCML 文件轉換爲同等的 XPCML 文件。這也讓您不必針對現有的 iSeries 程式呼叫文件撰寫 XPCML 來源檔，就能利用新的 XPCML 功能。

延伸與自訂 XPCML 綱目

XPCML 是可延伸的，這表示您可以定義新的參數類型，來延伸那些由 XPCML 綱目指定的參數類型。壓縮 XPCML 能延伸 XPCML 綱目，建立新的資料類型， 以簡化並增進 XPCML 文件的可靠性及實用性。

使用 XPCML 的基本要求

「可延伸程式呼叫標記語言 (XPCML)」的工作站 Java 虛擬機器基本要求，與其他 IBM Toolbox for Java 的相同。

如需詳細資訊，請參閱下列網頁：

第 8 頁的『執行 IBM Toolbox for Java 應用程式的工作站基本要求』

此外，若要使用 XPCML，還必須具備適合的 XML 綱目檔案以及「可延伸樣式表語言轉換」(Extensible Stylesheet Language Transformation, XSLT) 處理器：

XML 綱目檔案

您的 XPCML 文件必須知道綱目所在的檔案位於何處。IBM Toolbox for Java 的預設綱目是 xpcml.xsd，位於 jt400.jar 檔案中。

若要使用預設綱目，請從 jt400.jar 中取出 xpcml.xsd，然後將該檔案放到適當的位置。下列程序所說明的方法，可讓您取出工作站上的 .xsd 檔案。

取出 xpcml.xsd 綱目檔

- 從包含 jt400.jar 的目錄中的指令行階段作業開始
- 使用下列指令來取出 .xsd 檔案：

```
jar xvf  jt400.jar   com/ibm/as400/data/xpcml.xsd
```

註: 如果您不從包含 jt400.jar 的目錄中執行先前的指令，則可以指定指向 jt400.jar 的完整路徑。

您可以將預設綱目檔案 (或任何綱目檔案) 放在任何目錄中。 唯一的要求是，您必須要使用 <xpcml> 標籤中的 xsi:noNamespaceSchemaLocation 屬性來指定綱目檔案的位置。

您可以將綱目的位置指定爲檔案路徑或指定爲 URL。

註: 雖然下列範例使用 xpcml.xsd 作爲綱目檔案，但您還是可以指定可延伸 xpcml.xsd 的任何綱目。

- 若要指定與 XPCML 檔案相同的目錄，請使用 xsi:noNamespaceSchemaLocation='xpcml.xsd'
- 若要指定完整的路徑：xsi:noNamespaceSchemaLocation='c:\myDir\xpcml.xsd'
- 若要指定 URL：xsi:noNamespaceSchemaLocation='http://myServer/xpcml.xsd'

若要查看 HTML 版的 xpcml.xsd 檔案，請參閱下列頁面：

第 375 頁的『綱目 xpcml.xsd 檔案』

XML 剖析器與 XSLT 處理器

在執行時間，您必須在您的 CLASSPATH 環境變數中納入 XML 剖析器與 XSLT 處理器。如需詳細資訊，請參閱下列網頁：

第 370 頁的『XML 剖析器與 XSLT 處理器』

XPCML 綱目及語法

XPCML 文件 (稱爲 XPCML 來源檔) 中所包含的標籤與資料，可針對您 iSeries 伺服器上的程式完整地定義其呼叫。

由於 XPCML 使用 XML 綱目而不使用文件類型定義 (DTD)， 因此您可透過 PCML 無法使用的方式來使用 XPCML：

- 傳送輸入參數的值到您的程式作爲 XML 元素
- 從您的程式接收輸出參數作爲 XML 元素
- 讓 XML 剖析器自動驗證傳到您程式中的值
- 延伸綱目以定義新的簡式以及複合元素

如需有關 XPCML 綱目及語法的詳細資訊，請參閱下列頁面：

XPCML 來源與 PCML 來源的比較
檢查 XPCML 來源與 PCML 來源進行比較的範例。範例會說明 XPCML 如何提供更多的功能， 並使得原始資料變得更容易讀寫。

XPCML 綱目
檢查 XPCML 綱目檔案，瞭解更多有關使用及延伸 XPCML 綱目的資訊。

XPCML 語法
檢視綱目用來定義 XPCML 元素的 XPCML 語法元素清單。

XPCML 標籤屬性
說明 XPCML 綱目所定義之每一個元素的不同屬性。

XPCML 來源與 PCML 來源的比較:

XPCML 與 PCML 有多項差異，但一個主要的差異是，XPCML 可讓您在 XPCML 來源檔中指定輸入參數的值。

PCML 可讓您使用 <data> 標籤的 init 屬性來指定 PCML 來源中的資料元素起始值。然而，使用 PCML 來指定值有下列限制：

- 您無法使用 init 屬性來設定陣列值
- 對於 init 值的驗證只能在完成 PCML 文件的剖析後才能進行

若要在 PCML 中指定陣列值，您必須先讀取並剖析 PCML 文件，然後才能對 ProgramCallDocument.setValue() 執行一系列呼叫。

使用 XPCML 能讓您更易於指定單一元素及陣列的值：

- 在 XPCML 來源檔中指定純量與陣列元素兩者的值
- 在剖析時驗證指定的陣列值

透過下列簡單的比較來指出 XPCML 與 PCML 的差異處。每一個範例都會針對 iSeries 伺服器程式定義程式呼叫。

範例：呼叫 iSeries 伺服器程式

下列範例會呼叫名爲 prog1 的 iSeries 伺服器程式。

XPCML 來源碼

```
<?xml version="1.0" encoding="UTF-8"?>
<xpcml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation='xpcml.xsd' version="4.0">
    <program name="prog1" path="/QSYS.LIB/W95LIB.LIB/PROG1.PGM">
      <parameterList>
         <stringParm name="parm1"  passDirection="in" length="10">Parm1</stringParm>
         <intParm    name="parm2"  passDirection="in">5</intParm>
         <shortParm  name="parm3"  passDirection="in">3</shortParm>
      </parameterList>
    </program>
</xpcml>
```

PCML 來源碼

```
<pcml version="4.0">
  <program name="prog1" path="QSYS.LIB/MYLIB.LIB/PROG1.PGM">
     <data name="parm1"  type="char" usage="input" length="10" init="Parm1"/>
     <data name="parm2"  type="int"  usage="input" length="4"  init="5"/>
     <data name="parm3"  type="int"  usage="input" length="2"  precision="16" init="3"/>
  </program>
</pcml>
```

範例：使用字串參數陣列來呼叫 iSeries 伺服器程式

下列範例將呼叫名爲 prog2 的 iSeries 伺服器程式，並且將 parm1 定義爲字串參數陣列。請注意 XPCML 的功能性：

- 起始設定陣列中每一個元素的值
- 將輸入值設定爲執行完整驗證的 XML 剖析器所能驗證的元素內容

您可以利用 XPCML 的這一功能性優勢，而不必撰寫任何 Java 程式碼。

PCML 無法符合 XPCML 的效能。PCML 無法起始設定陣列中每一個元素的值。PCML 無法在剖析時驗證 init 值。若要符合 XPCML 功能，您必須先讀取並剖析 PCML 文件，然後編寫 Java 應用程式的程式碼，以設定每一個陣列元素的值。您同時也必須撰寫程式碼以驗證參數。

XPCML 來源碼

```
<?xml version="1.0" encoding="UTF-8"?>
<xpcml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="xpcml.xsd" version="4.0">

    <program name="prog2" path="/QSYS.LIB/W95LIB.LIB/PROG2.PGM">
      <parameterList>
         <arrayOfStringParm name="parm1" passDirection="in"
            length="10" count="3">
             <i>Parm1-First value</i>
             <i>Parm1-Second value</i>
             <i>Parm1-Third value</i>
         </arrayOfStringParm>
         <longParm name="parm2"  passDirection="in">5</longParm>
         <zonedDecimalParm name="parm3"  passDirection="in"
            totalDigits="5" fractionDigits="2">32.56</zonedDecimalParm>
      </parameterList>
    </program>
</xpcml>
```

PCML 來源碼

```
<pcml version="4.0">
  <program name="prog2" path="QSYS.LIB/MYLIB.LIB/PROG2.PGM">
     <data name="parm1"  type="char"  usage="input" length="20" count="3"/>
     <data name="parm2"  type="int"   usage="input" length="8"  init="5"/>
     <data name="parm3"  type="zoned" usage="input" length="5"  precision="2" init="32.56"/>
  </program>
</pcml>
```

綱目 xpcml.xsd 檔案:

有關使用 xpcml.xsd 檔案的詳細資訊，請參閱使用 XPCML 的基本需求。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

爲了方便顯示以及列印，此 xpcml.xsd 的 HTML 版本中有些程式行會折到第二行。相同的程式行在原始的 xsd 檔中則顯示在單一行中。

```
<?xml version="1.0" encoding="UTF-8"?>

<!--////////////////////////////////////////////////////////////////////////////
//
// JTOpen (IBM Toolbox for Java - OSS version)
//
// Filename: xpcml.xsd
//
// The source code contained herein is licensed under the IBM Public License
// Version 1.0, which has been approved by the Open Source Initiative.
// Copyright (C) 1997-2003 International Business Machines Corporation and
// others. All rights reserved.
//
////////////////////////////////////////////////////////////////////////////-->

<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>

<xs:annotation>
 <xs:documentation>
  Schema for xpcml (eXtended Program Call Markup Language).
 </xs:documentation>
</xs:annotation>

<xs:element name="xpcml">
  <xs:complexType>
    <xs:sequence>
```

```
      <xs:element ref="structOrProgram" minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="version" use="required">
     <xs:simpleType>
       <xs:restriction base="xs:string">
         <xs:enumeration value="4.0"/>
       </xs:restriction>
     </xs:simpleType>
   </xs:attribute>
  </xs:complexType>

  <!-- Define key/keyref link between the name of a struct   -->
  <!-- and the struct attribute of a parameter field.        -->
  <xs:key name="StructKey">
    <xs:selector xpath="struct"/>
    <xs:field xpath="@name"/>
  </xs:key>
  <xs:keyref name="spRef" refer="StructKey">
     <xs:selector xpath="structParm" />
     <xs:field xpath="@struct" />
  </xs:keyref>
</xs:element>

<!-- Program tag and attributes -->
<xs:element name="program" substitutionGroup="structOrProgram">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="parameterList" minOccurs="1" maxOccurs="1"/>
         <!-- Used as a wrapper tag around the parameter list for the program. -->
    </xs:sequence>
    <!-- Name of the program to call. -->
    <xs:attribute name="name" type="string50" use="required" />
    <!-- Path to the program object. Default is to assume in library QSYS. -->
    <xs:attribute name="path" type="xs:string"/>
    <!-- Specifies the order in which parameters should be parsed. -->
    <!-- Value is a blank-separated list of parameter names. -->
    <xs:attribute name="parseOrder" type="xs:string"/>
    <!-- The entry point name within a service program. -->
    <xs:attribute name="entryPoint" type="xs:string"/>
    <!-- The type of value, if any, returned from a service program call. -->
    <xs:attribute name="returnValue" type="returnValueType"/>
    <!-- When calling a Java program and iSeries program is on same server -->
    <!-- and is thread-safe, set to true to call the iSeries program in same job -->
    <!-- and on same thread as the Java program. -->
    <xs:attribute name="threadSafe" type="xs:boolean"  />
    <!-- The CCSID of the entry point name within a service program. -->
    <xs:attribute name="epccsid" type="ccsidType"/>
  </xs:complexType>
</xs:element>

<!-- A parameter list is made up of one or more parameters. -->
<xs:element name="parameterList">
  <xs:complexType>
      <xs:group ref="programParameter" minOccurs="1" maxOccurs="unbounded"/>
   </xs:complexType>
</xs:element>

<!-- All the different kinds of program parameters that we understand. -->
<xs:group name="programParameter">
      <xs:choice>
 <xs:element ref="stringParmGroup"/>
 <xs:element ref="stringParmArrayGroup"/>
 <xs:element ref="intParmGroup"/>
 <xs:element ref="intParmArrayGroup"/>
        <xs:element ref="unsignedIntParmGroup"/>
        <xs:element ref="unsignedIntParmArrayGroup"/>
 <xs:element ref="shortParmGroup"/>
```

```
 <xs:element ref="shortParmArrayGroup"/>
 <xs:element ref="unsignedShortParmGroup"/>
 <xs:element ref="unsignedShortParmArrayGroup"/>
 <xs:element ref="longParmGroup"/>
 <xs:element ref="longParmArrayGroup"/>
 <xs:element ref="zonedDecimalParmGroup"/>
 <xs:element ref="zonedDecimalParmArrayGroup"/>
 <xs:element ref="packedDecimalParmGroup"/>
        <xs:element ref="packedDecimalParmArrayGroup"/>
 <xs:element ref="floatParmGroup"/>
 <xs:element ref="floatParmArrayGroup"/>
 <xs:element ref="doubleParmGroup"/>
 <xs:element ref="doubleParmArrayGroup"/>
 <xs:element ref="hexBinaryParmGroup"/>
 <xs:element ref="hexBinaryParmArrayGroup"/>
        <xs:element ref="structParmGroup"/>
        <xs:element ref="structParmArrayGroup"/>
        <xs:element ref="structArrayGroup"/>
        <xs:element ref="struct"/>
      </xs:choice>

</xs:group>

<!-- Abstract type for all data parameter types. -->
<xs:element name="stringParmGroup" type="stringParmType" abstract="true" />
<xs:element name="stringParmArrayGroup" type="stringParmArrayType" abstract="true" />
<xs:element name="intParmGroup" type="intParmType" abstract="true" />
<xs:element name="intParmArrayGroup" type="intParmArrayType" abstract="true" />
<xs:element name="unsignedIntParmGroup" type="unsignedIntParmType" abstract="true" />
<xs:element name="unsignedIntParmArrayGroup" type="unsignedIntParmArrayType" abstract="true" />
<xs:element name="shortParmGroup" type="shortParmType" abstract="true" />
<xs:element name="shortParmArrayGroup" type="shortParmArrayType" abstract="true" />
<xs:element name="unsignedShortParmGroup" type="unsignedShortParmType" abstract="true" />
<xs:element name="unsignedShortParmArrayGroup" type="unsignedShortParmArrayType" abstract="true" />
<xs:element name="longParmGroup" type="longParmType" abstract="true" />
<xs:element name="longParmArrayGroup" type="longParmArrayType" abstract="true" />
<xs:element name="zonedDecimalParmGroup" type="zonedDecimalParmType" abstract="true" />
<xs:element name="zonedDecimalParmArrayGroup" type="zonedDecimalParmArrayType" abstract="true" />
<xs:element name="packedDecimalParmGroup" type="packedDecimalParmType" abstract="true" />
<xs:element name="packedDecimalParmArrayGroup" type="packedDecimalParmArrayType" abstract="true" />
<xs:element name="floatParmGroup" type="floatParmType" abstract="true" />
<xs:element name="floatParmArrayGroup" type="floatParmArrayType" abstract="true" />
<xs:element name="doubleParmGroup" type="doubleParmType" abstract="true" />
<xs:element name="doubleParmArrayGroup" type="doubleParmArrayType" abstract="true" />
<xs:element name="hexBinaryParmGroup" type="hexBinaryParmType" abstract="true" />
<xs:element name="hexBinaryParmArrayGroup" type="hexBinaryParmArrayType" abstract="true" />
<xs:element name="structParmGroup" type="structParmType" abstract="true" />
<xs:element name="structParmArrayGroup" type="structParmArrayType" abstract="true" />
<xs:element name="structArrayGroup" type="structArrayType" abstract="true"
            substitutionGroup="structOrProgram" />

<!-- String parameter -->
   <xs:element name="stringParm" type="stringParmType" substitutionGroup="stringParmGroup"
               nillable="true"/>
     <xs:complexType name="stringParmType">
       <xs:simpleContent>
          <xs:extension base="stringFieldType">
            <xs:attributeGroup ref="commonParmAttrs"/>
          </xs:extension>
       </xs:simpleContent>
   </xs:complexType>

<!-- Array of string parameters -->
  <xs:element name="arrayOfStringParm" type="stringParmArrayType"
              substitutionGroup="stringParmArrayGroup" nillable="true" />
  <xs:complexType name="stringParmArrayType">
    <xs:sequence>
```

```
      <xs:element name="i" type="stringElementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attributeGroup ref="commonParmAttrs"/>
    <xs:attributeGroup ref="commonFieldAttrs"/>
    <!-- The number of elements in the array. -->
    <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
    <xs:attribute name="count" type="xs:string" />
    <!-- The number of characters in each string. -->
    <xs:attribute name="length" type="xs:string"/>
    <!-- The host CCSID for each string. -->
    <xs:attribute name="ccsid" type="xs:string"/>
    <!-- Specifies how to trim whitespace (left, right, both, none). -->
    <xs:attribute name="trim" type="trimType" />
    <!-- The size of each character ('chartype' in PCML). -->
    <xs:attribute name="bytesPerChar" type="charType" />
   <!-- The bidirectional string type. -->
    <xs:attribute name="bidiStringType" type="bidiStringTypeType" />
  </xs:complexType>

      <xs:complexType name="stringElementType">
        <xs:simpleContent>
           <xs:extension base="xs:string">
                <!-- The index into the array. -->
                <xs:attribute name="index" type="xs:nonNegativeInteger"  />
           </xs:extension>
        </xs:simpleContent>
       </xs:complexType>


<!-- Integer parameter (4 bytes on server) -->
    <xs:element name="intParm" type="intParmType"  nillable="true" substitutionGroup="intParmGroup"   />
      <xs:complexType name="intParmType" >
         <xs:simpleContent>
            <xs:extension base="intFieldType">
              <xs:attributeGroup ref="commonParmAttrs"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>

<!-- intParm array type -->
  <xs:element name="arrayOfIntParm" type="intParmArrayType" substitutionGroup="intParmArrayGroup"
              nillable="true" />
  <xs:complexType name="intParmArrayType">
    <xs:sequence>
      <!-- 'i' is the tag used for non-struct array elements. -->
      <xs:element name="i" type="intElementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
    <xs:attribute name="count" type="xs:string" />
    <xs:attributeGroup ref="commonParmAttrs"/>
    <xs:attributeGroup ref="commonFieldAttrs"/>
  </xs:complexType>

  <xs:complexType name="intElementType">
    <xs:simpleContent>
       <xs:extension base="xs:int">
            <xs:attribute name="index" type="xs:nonNegativeInteger"  />
       </xs:extension>
    </xs:simpleContent>
  </xs:complexType>


<!-- Unsigned Integer parameter (4 bytes on server) -->
    <xs:element name="unsignedIntParm" type="unsignedIntParmType" nillable="true"
                substitutionGroup="unsignedIntParmGroup"   />
      <xs:complexType name="unsignedIntParmType">
         <xs:simpleContent>
```

```
            <xs:extension base="unsignedIntFieldType">
              <xs:attributeGroup ref="commonParmAttrs"/>
            </xs:extension>
          </xs:simpleContent>
      </xs:complexType>

<!-- unsigned intParm array type -->
  <xs:element name="arrayOfUnsignedIntParm" type="unsignedIntParmArrayType"
              substitutionGroup="unsignedIntParmArrayGroup" nillable="true" />
  <xs:complexType name="unsignedIntParmArrayType">
    <xs:sequence>
      <xs:element name="i" type="unsignedIntElementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
    <xs:attribute name="count" type="xs:string" />
    <xs:attributeGroup ref="commonParmAttrs"/>
    <xs:attributeGroup ref="commonFieldAttrs"/>
  </xs:complexType>

  <xs:complexType name="unsignedIntElementType">
    <xs:simpleContent>
       <xs:extension base="xs:unsignedInt">
            <xs:attribute name="index" type="xs:nonNegativeInteger"  />
       </xs:extension>
    </xs:simpleContent>
  </xs:complexType>


<!-- Short integer parameter (2 bytes on server) -->
   <xs:element name="shortParm" type="shortParmType" nillable="true" substitutionGroup="shortParmGroup"/>
      <xs:complexType name="shortParmType">
         <xs:simpleContent>
            <xs:extension base="shortFieldType">
              <xs:attributeGroup ref="commonParmAttrs"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>

<!-- shortParm array type -->
  <xs:element name="arrayOfShortParm" type="shortParmArrayType" substitutionGroup="shortParmArrayGroup"
              nillable="true" />
  <xs:complexType name="shortParmArrayType">
    <xs:sequence>
      <xs:element name="i" type="shortElementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
    <xs:attribute name="count" type="xs:string" />
    <xs:attributeGroup ref="commonParmAttrs"/>
    <xs:attributeGroup ref="commonFieldAttrs"/>
  </xs:complexType>

  <xs:complexType name="shortElementType">
    <xs:simpleContent>
       <xs:extension base="xs:short">
            <xs:attribute name="index" type="xs:nonNegativeInteger"  />
       </xs:extension>
    </xs:simpleContent>
  </xs:complexType>


<!-- Unsigned Short integer parameter (2 bytes on server) -->
    <xs:element name="unsignedShortParm" type="unsignedShortParmType" nillable="true"
                substitutionGroup="unsignedShortParmGroup"   />
      <xs:complexType name="unsignedShortParmType">
         <xs:simpleContent>
            <xs:extension base="unsignedShortFieldType">
              <xs:attributeGroup ref="commonParmAttrs"/>
```

```
            </xs:extension>
          </xs:simpleContent>
      </xs:complexType>

<!-- unsignedShortParm array type -->
  <xs:element name="arrayOfUnsignedShortParm" type="unsignedShortParmArrayType"
              substitutionGroup="unsignedShortParmArrayGroup" nillable="true" />
  <xs:complexType name="unsignedShortParmArrayType">
    <xs:sequence>
      <xs:element name="i" type="unsignedShortElementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
    <xs:attribute name="count" type="xs:string" />
    <xs:attributeGroup ref="commonParmAttrs"/>
    <xs:attributeGroup ref="commonFieldAttrs"/>
  </xs:complexType>

  <xs:complexType name="unsignedShortElementType">
    <xs:simpleContent>
       <xs:extension base="xs:unsignedShort">
            <xs:attribute name="index" type="xs:nonNegativeInteger"  />
       </xs:extension>
    </xs:simpleContent>
  </xs:complexType>


<!-- Long integer parameter (8 bytes on server) -->
    <xs:element name="longParm" type="longParmType" nillable="true" substitutionGroup="longParmGroup"  />
      <xs:complexType name="longParmType">
         <xs:simpleContent>
            <xs:extension base="longFieldType">
              <xs:attributeGroup ref="commonParmAttrs"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>

<!-- longParm array type -->
  <xs:element name="arrayOfLongParm" type="longParmArrayType" substitutionGroup="longParmArrayGroup"
              nillable="true" />
  <xs:complexType name="longParmArrayType">
    <xs:sequence>
      <xs:element name="i" type="longElementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
    <xs:attribute name="count" type="xs:string" />
    <xs:attributeGroup ref="commonParmAttrs"/>
    <xs:attributeGroup ref="commonFieldAttrs"/>
  </xs:complexType>

  <xs:complexType name="longElementType">
    <xs:simpleContent>
       <xs:extension base="xs:long">
            <xs:attribute name="index" type="xs:nonNegativeInteger"  />
       </xs:extension>
    </xs:simpleContent>
  </xs:complexType>


<!-- ZonedDecimal parameter -->
    <xs:element name="zonedDecimalParm" type="zonedDecimalParmType" nillable="true"
                substitutionGroup="zonedDecimalParmGroup"   />
      <xs:complexType name="zonedDecimalParmType">
         <xs:simpleContent>
            <xs:extension base="zonedDecimalFieldType">
              <xs:attributeGroup ref="commonParmAttrs"/>
            </xs:extension>
         </xs:simpleContent>
```

```
      </xs:complexType>

<!-- zonedDecimalParm array type -->
  <xs:element name="arrayOfZonedDecimalParm" type="zonedDecimalParmArrayType"
              substitutionGroup="zonedDecimalParmArrayGroup" nillable="true" />
  <xs:complexType name="zonedDecimalParmArrayType">
    <xs:sequence>
      <xs:element name="i" type="zonedDecimalElementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attributeGroup ref="commonParmAttrs"/>
    <xs:attributeGroup ref="commonFieldAttrs"/>
    <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
    <xs:attribute name="count" type="xs:string" />
    <!-- The total number of digits in the field ('length' in PCML). -->
    <xs:attribute name="totalDigits" type="xs:positiveInteger"/>
    <!-- The number of fractional digits ('precision' in PCML). -->
    <xs:attribute name="fractionDigits" type="xs:nonNegativeInteger"/>
  </xs:complexType>

  <xs:complexType name="zonedDecimalElementType">
    <xs:simpleContent>
       <xs:extension base="xs:decimal">
            <xs:attribute name="index" type="xs:nonNegativeInteger"  />
       </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

<!-- packedDecimal parameter -->
    <xs:element name="packedDecimalParm" type="packedDecimalParmType" nillable="true"
                substitutionGroup="packedDecimalParmGroup"   />
      <xs:complexType name="packedDecimalParmType">
         <xs:simpleContent>
            <xs:extension base="packedDecimalFieldType">
              <xs:attributeGroup ref="commonParmAttrs"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>

<!-- packedDecimalParm array type -->
  <xs:element name="arrayOfPackedDecimalParm" type="packedDecimalParmArrayType"
              substitutionGroup="packedDecimalParmArrayGroup" nillable="true" />
  <xs:complexType name="packedDecimalParmArrayType">
    <xs:sequence>
      <xs:element name="i" type="packedDecimalElementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attributeGroup ref="commonParmAttrs"/>
    <xs:attributeGroup ref="commonFieldAttrs"/>
    <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
    <xs:attribute name="count" type="xs:string" />
    <xs:attribute name="totalDigits" type="xs:positiveInteger"/>
    <xs:attribute name="fractionDigits" type="xs:nonNegativeInteger"/>
  </xs:complexType>

  <xs:complexType name="packedDecimalElementType">
    <xs:simpleContent>
       <xs:extension base="xs:decimal">
            <xs:attribute name="index" type="xs:nonNegativeInteger"  />
       </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

<!-- Float parameter (4 bytes on server) -->
   <xs:element name="floatParm" type="floatParmType" nillable="true" substitutionGroup="floatParmGroup"/>
      <xs:complexType name="floatParmType">
         <xs:simpleContent>
            <xs:extension base="floatFieldType">
              <xs:attributeGroup ref="commonParmAttrs"/>
```

```
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>

<!-- floatParm array type -->
  <xs:element name="arrayOfFloatParm" type="floatParmArrayType" substitutionGroup="floatParmArrayGroup"
              nillable="true" />
  <xs:complexType name="floatParmArrayType">
    <xs:sequence>
      <xs:element name="i" type="floatElementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
    <xs:attribute name="count" type="xs:string" />
    <xs:attributeGroup ref="commonParmAttrs"/>
    <xs:attributeGroup ref="commonFieldAttrs"/>
  </xs:complexType>

  <xs:complexType name="floatElementType">
    <xs:simpleContent>
       <xs:extension base="xs:float">
            <xs:attribute name="index" type="xs:nonNegativeInteger"  />
       </xs:extension>
    </xs:simpleContent>
  </xs:complexType>


<!-- Double parameter (8 bytes on server) -->
    <xs:element name="doubleParm" type="doubleParmType" nillable="true"
                substitutionGroup="doubleParmGroup"   />
      <xs:complexType name="doubleParmType">
         <xs:simpleContent>
            <xs:extension base="doubleFieldType">
              <xs:attributeGroup ref="commonParmAttrs"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>

<!-- doubleParm array type -->
  <xs:element name="arrayOfDoubleParm" type="doubleParmArrayType"
              substitutionGroup="doubleParmArrayGroup" nillable="true" />
  <xs:complexType name="doubleParmArrayType">
    <xs:sequence>
      <xs:element name="i" type="doubleElementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
    <xs:attribute name="count" type="xs:string" />
    <xs:attributeGroup ref="commonParmAttrs"/>
    <xs:attributeGroup ref="commonFieldAttrs"/>
  </xs:complexType>

  <xs:complexType name="doubleElementType">
    <xs:simpleContent>
       <xs:extension base="xs:double">
            <xs:attribute name="index" type="xs:nonNegativeInteger"  />
       </xs:extension>
    </xs:simpleContent>
  </xs:complexType>


<!-- Hex binary parameter (any number of bytes; unsigned) -->
    <xs:element name="hexBinaryParm" type="hexBinaryParmType"  substitutionGroup="hexBinaryParmGroup"  />
      <xs:complexType name="hexBinaryParmType">
         <xs:simpleContent>
            <xs:extension base="hexBinaryFieldType">
              <!-- The field length in bytes ('length' in PCML). -->
              <xs:attribute name="totalBytes" type="xs:string"/>
              <xs:attributeGroup ref="commonParmAttrs"/>
```

```
            </xs:extension>
          </xs:simpleContent>
       </xs:complexType>

<!-- hexBinaryParm array type -->
  <xs:element name="arrayOfHexBinaryParm" type="hexBinaryParmArrayType"
              substitutionGroup="hexBinaryParmArrayGroup" nillable="true" />
  <xs:complexType name="hexBinaryParmArrayType">
    <xs:sequence>
      <xs:element name="i" type="hexBinaryElementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="totalBytes" type="xs:string"/>
    <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
    <xs:attribute name="count" type="xs:string" />
    <xs:attributeGroup ref="commonParmAttrs"/>
    <xs:attributeGroup ref="commonFieldAttrs"/>
  </xs:complexType>

  <xs:complexType name="hexBinaryElementType">
    <xs:simpleContent>
       <xs:extension base="xs:hexBinary">
            <xs:attribute name="index" type="xs:nonNegativeInteger"  />
       </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

<!-- Structure parm type -->
  <xs:element name="structParm" type="structParmType"  substitutionGroup="structParmGroup"   />
      <xs:complexType name="structParmType">
         <xs:complexContent>
            <xs:extension base="structureParmArray">
              <xs:attribute name="struct" type="string50"/>
              <!-- Specifies whether the parameter is passed by value or reference ('passby' in PCML).-->
              <!-- Value only allowed for integer parameters. -->
              <xs:attribute name="passMode" type="passModeType"/>
              <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
              <xs:attribute name="count" type="xs:string"/>
            </xs:extension>
         </xs:complexContent>
      </xs:complexType>

<!-- Structure parm array type -->
  <xs:element name="arrayOfStructParm" type="structParmArrayType"
              substitutionGroup="structParmArrayGroup" nillable="true" />
  <xs:complexType name="structParmArrayType">
    <xs:sequence>
        <!-- struct_i tag represents struct or struct parm array elements. -->
        <xs:element name="struct_i" type="structElementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
    <xs:attribute name="count" type="xs:string" />
    <xs:attributeGroup ref="commonParmAttrs"/>
    <xs:attributeGroup ref="commonFieldAttrs"/>
    <xs:attribute name="struct" type="string50"/>
  </xs:complexType>

  <xs:complexType name="structElementType">
    <xs:complexContent>
       <xs:extension base="structureParmArray">
            <xs:attribute name="index" type="xs:nonNegativeInteger"  />
       </xs:extension>
    </xs:complexContent>
  </xs:complexType>


<!-- Struct element -->
  <xs:element name="struct" type="structureParmArray" substitutionGroup="structOrProgram" />
```

```
<!-- Struct array type -->
  <xs:element name="arrayOfStruct" type="structArrayType" substitutionGroup="structArrayGroup"
              nillable="true" />
  <xs:complexType name="structArrayType">
    <xs:sequence>
       <!-- struct_i tag represents struct elements in an array. -->
       <xs:element name="struct_i" type="structElementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <!-- The name of the struct. -->
    <xs:attribute name="name" type="string50"/>
    <!-- Number of elements in the array. -->
    <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
    <xs:attribute name="count" type="xs:string" />
    <!-- Specifies whether this is an input, output, or input-output struct ('usage' in PCML). -->
    <xs:attribute name="passDirection" type="passDirectionType"/>
    <!-- The offset to the struct within an output parameter. -->
    <xs:attribute name="offset" type="xs:string" />
    <!-- The base location from which the 'offset' attribute is relative. -->
    <xs:attribute name="offsetFrom" type="xs:string" />
    <!-- The number of bytes to reserve for output data for the element. -->
    <xs:attribute name="outputSize" type="xs:string" />
    <!-- The lowest version of i5/OS on which this element exists. -->
    <xs:attribute name="minvrm" type="string10" />
    <!-- The highest version of i5/OS on which this element exists. -->
    <xs:attribute name="maxvrm" type="string10" />
  </xs:complexType>


<!-- Attributes that are common to all data field types. -->
  <xs:attributeGroup name="commonParmAttrs">
     <!-- Specifies whether this is an input, output, or input-output parameter ('usage' in PCML). -->
     <!-- The default value if none is specified is 'inherit'. -->
     <xs:attribute name="passDirection" type="passDirectionType"/>
     <!-- Specifies whether the parameter is passed by reference or value ('passby' in PCML). -->
     <!-- The default value if none is specified is 'reference'. -->
     <xs:attribute name="passMode" type="passModeType" />
     <!-- The offset to the element within an output parameter. -->
     <!-- The default value if none is specified is 0. -->
     <xs:attribute name="offset" type="xs:string" />
     <!-- The base location from which the 'offset' attribute is relative. -->
     <xs:attribute name="offsetFrom" type="xs:string" />
     <!-- The number of bytes to reserve for output data for the element. -->
     <xs:attribute name="outputSize" type="xs:string" />
     <!-- The lowest version of i5/OS to which this field applies. -->
     <!-- If not specified, we assume this field applies to all versions. -->
     <xs:attribute name="minvrm" type="string10" />
     <!-- The highest version of i5/OS to which this field applies. -->
     <!-- If not specified, we assume this field applies to all versions. -->
     <xs:attribute name="maxvrm" type="string10" />
  </xs:attributeGroup>

  <xs:simpleType name="passDirectionType">
    <xs:restriction base="xs:string">
       <xs:enumeration value="in"/>
       <xs:enumeration value="inout"/>
       <xs:enumeration value="out"/>
       <xs:enumeration value="inherit"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="passModeType">
    <xs:restriction base="xs:string">
       <xs:enumeration value="value"/>
       <xs:enumeration value="reference"/>
     </xs:restriction>
  </xs:simpleType>
```

```
  <!-- Following types are to maintain compatibility with PCML -->
  <xs:simpleType name="bidiStringTypeType">
    <xs:restriction base="xs:string">
       <xs:enumeration value="ST4"/>
       <xs:enumeration value="ST5"/>
       <xs:enumeration value="ST6"/>
       <xs:enumeration value="ST7"/>
       <xs:enumeration value="ST8"/>
       <xs:enumeration value="ST9"/>
       <xs:enumeration value="ST10"/>
       <xs:enumeration value="ST11"/>
       <xs:enumeration value="DEFAULT"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="charType">
    <xs:restriction base="xs:string">
       <xs:enumeration value="onebyte"/>
       <xs:enumeration value="twobyte"/>
     </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="trimType">
    <xs:restriction base="xs:string">
       <xs:enumeration value="none"/>
       <xs:enumeration value="left"/>
       <xs:enumeration value="right"/>
       <xs:enumeration value="both"/>
     </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="returnValueType">
    <xs:restriction base="xs:string">
       <xs:enumeration value="void"/>
       <xs:enumeration value="integer"/>
     </xs:restriction>
  </xs:simpleType>


  <xs:complexType name="structureParmArray">
    <xs:sequence>
       <xs:group ref="structureParm" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="string50"/>
    <xs:attribute name="passDirection" type="passDirectionType"/>
    <xs:attribute name="offset" type="xs:string" />
    <xs:attribute name="offsetFrom" type="xs:string" />
    <xs:attribute name="outputSize" type="xs:string" />
    <xs:attribute name="minvrm" type="string10" />
    <xs:attribute name="maxvrm" type="string10" />
  </xs:complexType>


<!-- A structureParm is exactly one of the following: stringParm, intParm,
     shortParm, longParm, zonedDecimalParm, packedDecimalParm, floatParm,
     doubleParm, or hexBinaryParm. -->
      <xs:group name="structureParm">
        <xs:choice>
          <xs:element ref="stringParmGroup"  />
          <xs:element ref="stringParmArrayGroup"  />
          <xs:element ref="intParmGroup"   />
          <xs:element ref="intParmArrayGroup"   />
          <xs:element ref="unsignedIntParmGroup"   />
          <xs:element ref="unsignedIntParmArrayGroup"   />
          <xs:element ref="shortParmGroup" />
          <xs:element ref="shortParmArrayGroup" />
```

```
          <xs:element ref="unsignedShortParmGroup"   />
          <xs:element ref="unsignedShortParmArrayGroup"   />
          <xs:element ref="longParmGroup"  />
          <xs:element ref="longParmArrayGroup"  />
          <xs:element ref="zonedDecimalParmGroup"  />
          <xs:element ref="zonedDecimalParmArrayGroup"  />
          <xs:element ref="packedDecimalParmGroup" />
          <xs:element ref="packedDecimalParmArrayGroup" />
          <xs:element ref="floatParmGroup" />
          <xs:element ref="floatParmArrayGroup" />
          <xs:element ref="doubleParmGroup" />
          <xs:element ref="doubleParmArrayGroup" />
          <xs:element ref="hexBinaryParmGroup" />
          <xs:element ref="hexBinaryParmArrayGroup" />
          <xs:element ref="structParmGroup" />
          <xs:element ref="structParmArrayGroup"/>
          <xs:element ref="structArrayGroup"/>
          <xs:element ref="struct"/>
        </xs:choice>
      </xs:group>


<!-- Field Definition schema. -->

  <!-- Define basic iSeries native data types. -->

     <xs:complexType name="zonedDecimal">
        <xs:simpleContent>
           <xs:extension base="xs:decimal">
                <xs:attribute name="totalDigits" type="xs:positiveInteger"  />
                <xs:attribute name="fractionDigits" type="xs:nonNegativeInteger" />
           </xs:extension>
        </xs:simpleContent>
     </xs:complexType>


     <xs:complexType name="packedDecimal">
        <xs:simpleContent>
           <xs:extension base="xs:decimal">
                <xs:attribute name="totalDigits" type="xs:positiveInteger" />
                <xs:attribute name="fractionDigits" type="xs:nonNegativeInteger" />
           </xs:extension>
        </xs:simpleContent>
     </xs:complexType>


    <xs:complexType name="structureFieldArray">
     <xs:sequence>
        <xs:group ref="structureField" minOccurs="1" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:attribute name="name" type="string50"/>
     <!-- 'count' is required if you want to input and/or output array data as XPCML. -->
     <xs:attribute name="count" type="xs:string"/>
  </xs:complexType>



  <!-- Abstract type for "struct or program". -->
  <xs:element name="structOrProgram" abstract="true" />


  <!-- Abstract type for all data field types. -->
  <xs:element name="stringFieldGroup" type="stringFieldType" abstract="true" />
  <xs:element name="intFieldGroup" type="intFieldType" abstract="true" />
  <xs:element name="unsignedIntFieldGroup" type="unsignedIntFieldType" abstract="true" />
  <xs:element name="shortFieldGroup" type="shortFieldType" abstract="true" />
  <xs:element name="unsignedShortFieldGroup" type="unsignedShortFieldType" abstract="true" />
```

```
  <xs:element name="longFieldGroup" type="longFieldType" abstract="true" />
  <xs:element name="zonedDecimalFieldGroup" type="zonedDecimalFieldType" abstract="true" />
  <xs:element name="packedDecimalFieldGroup" type="packedDecimalFieldType" abstract="true" />
  <xs:element name="floatFieldGroup" type="floatFieldType" abstract="true" />
  <xs:element name="doubleFieldGroup" type="doubleFieldType" abstract="true" />
  <xs:element name="hexBinaryFieldGroup" type="hexBinaryFieldType" abstract="true" />
  <xs:element name="structFieldGroup" type="structFieldType" abstract="true" />



  <!-- Declare each field element to be a specific field type. -->
  <xs:element name="stringField" type="stringFieldType" substitutionGroup="stringFieldGroup"
              nillable="true"/>
  <xs:element name="intField" type="intFieldType" nillable="true"
              substitutionGroup="intFieldGroup" />
  <xs:element name="unsignedIntField" type="unsignedIntFieldType"
              substitutionGroup="unsignedIntFieldGroup" nillable="true"/>
  <xs:element name="shortField" type="shortFieldType" nillable="true"
              substitutionGroup="shortFieldGroup" />
  <xs:element name="unsignedShortField" type="unsignedShortFieldType" nillable="true"
              substitutionGroup="unsignedShortFieldGroup"  />
  <xs:element name="longField" type="longFieldType" nillable="true"
              substitutionGroup="longFieldGroup" />
  <xs:element name="hexBinaryField" type="hexBinaryFieldType" nillable="true"
              substitutionGroup="hexBinaryFieldGroup" />
  <xs:element name="zonedDecimalField" type="zonedDecimalFieldType" nillable="true"
              substitutionGroup="zonedDecimalFieldGroup"  />
  <xs:element name="packedDecimalField" type="packedDecimalFieldType" nillable="true"
              substitutionGroup="packedDecimalFieldGroup" />
  <xs:element name="doubleField" type="doubleFieldType" nillable="true"
              substitutionGroup="doubleFieldGroup" />
  <xs:element name="floatField" type="floatFieldType" nillable="true"
              substitutionGroup="floatFieldGroup" />


  <xs:element name="structField" type="structFieldType" nillable="true"
              substitutionGroup="structFieldGroup" />

<!-- A StructureField is exactly one of the following: stringField, intField,
     shortField, longField, zonedDecimalField, packedDecimalField, floatField,
     doubleField, or hexBinaryField. -->
      <xs:group name="structureField">
        <xs:choice>
          <xs:element ref="stringFieldGroup"/>
          <xs:element ref="intFieldGroup"/>
          <xs:element ref="unsignedIntFieldGroup"/>
          <xs:element ref="shortFieldGroup"/>
          <xs:element ref="unsignedShortFieldGroup"/>
          <xs:element ref="longFieldGroup"/>
          <xs:element ref="zonedDecimalFieldGroup"/>
          <xs:element ref="packedDecimalFieldGroup"/>
          <xs:element ref="floatFieldGroup"/>
          <xs:element ref="doubleFieldGroup"/>
          <xs:element ref="hexBinaryFieldGroup"/>
          <xs:element ref="structParmGroup"/>
          <xs:element ref="struct"/>
        </xs:choice>
      </xs:group>

  <!-- Character field -->
  <!-- Maps to AS400Text. -->
   <xs:complexType name="stringFieldType">
     <xs:simpleContent>
        <xs:extension base="xs:string">
          <!-- Number of characters. -->
          <xs:attribute name="length" type="xs:string"/>
          <!-- Indicates the field's encoding (CCSID) on the server. -->
```

```
        <xs:attribute name="ccsid" type="xs:string"/>
        <xs:attribute name="trim" type="trimType" />
        <xs:attribute name="bytesPerChar" type="charType" />
        <xs:attribute name="bidiStringType" type="bidiStringTypeType" />
        <xs:attributeGroup ref="commonFieldAttrs"/>
      </xs:extension>
   </xs:simpleContent>
 </xs:complexType>


<!-- hexBinary field -->
<!-- Maps to AS400ByteArray. -->
 <xs:complexType name="hexBinaryFieldType">
   <xs:simpleContent>
      <xs:extension base="xs:hexBinary">
        <xs:attributeGroup ref="commonFieldAttrs"/>
      </xs:extension>
   </xs:simpleContent>
 </xs:complexType>


<!-- Float field -->
<!-- Maps to AS400Float4. -->
 <xs:complexType name="floatFieldType">
   <xs:simpleContent>
       <xs:extension base="xs:float">
           <xs:attributeGroup ref="commonFieldAttrs"/>
       </xs:extension>
   </xs:simpleContent>
  </xs:complexType>

<!-- zonedDecimal field -->
<!-- Maps to AS400ZonedDecimal. -->
 <xs:complexType name="zonedDecimalFieldType">
   <xs:simpleContent>
       <xs:extension base="zonedDecimal">
           <xs:attributeGroup ref="commonFieldAttrs"/>
       </xs:extension>
   </xs:simpleContent>
 </xs:complexType>

<!-- packedDecimal field -->
<!-- Maps to AS400PackedDecimal. -->
 <xs:complexType name="packedDecimalFieldType">
   <xs:simpleContent>
       <!-- In DDS, "binary" values are 1-18 digits; if field length is
            greater than 9, then decimal positions value must be 0. -->
       <xs:extension base="packedDecimal">
           <xs:attributeGroup ref="commonFieldAttrs"/>
       </xs:extension>
   </xs:simpleContent>
 </xs:complexType>

<!-- int field -->
<!-- Maps to AS400Bin4. -->
 <xs:complexType name="intFieldType">
   <xs:simpleContent>
       <xs:extension base="xs:int">
          <xs:attributeGroup ref="commonFieldAttrs"/>
       </xs:extension>
   </xs:simpleContent>
 </xs:complexType>

<!-- unsigned int field -->
<!-- Maps to AS400Bin4. -->
 <xs:complexType name="unsignedIntFieldType">
   <xs:simpleContent>
```

```
        <xs:extension base="xs:unsignedInt">
           <xs:attributeGroup ref="commonFieldAttrs"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>

   <!-- short field -->
   <!-- Maps to AS400Bin2. -->
    <xs:complexType name="shortFieldType">
      <xs:simpleContent>
          <xs:extension base="xs:short">
             <xs:attributeGroup ref="commonFieldAttrs"/>
          </xs:extension>
      </xs:simpleContent>
    </xs:complexType>

   <!-- unsigned short field -->
   <!-- Maps to AS400Bin2. -->
    <xs:complexType name="unsignedShortFieldType">
      <xs:simpleContent>
          <xs:extension base="xs:unsignedShort">
             <xs:attributeGroup ref="commonFieldAttrs"/>
          </xs:extension>
      </xs:simpleContent>
    </xs:complexType>


   <!-- long field -->
   <!-- Maps to AS400Bin8. -->
    <xs:complexType name="longFieldType">
      <xs:simpleContent>
          <xs:extension base="xs:long">
             <xs:attributeGroup ref="commonFieldAttrs"/>
          </xs:extension>
      </xs:simpleContent>
    </xs:complexType>

   <!-- double field -->
   <!-- Maps to AS400Float8. -->
    <xs:complexType name="doubleFieldType">
      <xs:simpleContent>
          <xs:extension base="xs:double">
             <xs:attributeGroup ref="commonFieldAttrs"/>
          </xs:extension>
      </xs:simpleContent>
    </xs:complexType>



   <!-- struct Field -->
    <xs:complexType name="structFieldType">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="struct" type="string50"/>
          <xs:attributeGroup ref="commonFieldAttrs"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>


<!-- Attributes that are common to all data field types. -->
<xs:attributeGroup name="commonFieldAttrs">
   <xs:attribute name="name" type="string50"/>
   <xs:attribute name="columnHeading1" type="string20"/>
   <xs:attribute name="columnHeading2" type="string20"/>
   <xs:attribute name="columnHeading3" type="string20"/>
   <xs:attribute name="description" type="string50"/>
```

```
   <xs:attribute name="defaultValue" type="xs:string"/><!-- Max length of string is 65535 characters. -->
   <xs:attribute name="nullable" type="xs:boolean"/>
   <xs:attribute name="isEmptyString" type="xs:boolean"/><!-- Indicate this is an empty string. -->
</xs:attributeGroup>

<!-- Utility types.  -->

<xs:simpleType name="ccsidType">
                     <xs:restriction base="xs:nonNegativeInteger">
                      <xs:maxInclusive value="65535"/>
                     </xs:restriction>
                    </xs:simpleType>

<xs:simpleType name="string10">
                     <xs:restriction base="xs:string">
                      <xs:maxLength value="10"/>
                     </xs:restriction>
                    </xs:simpleType>

<xs:simpleType name="string20">
                     <xs:restriction base="xs:string">
                      <xs:maxLength value="20"/>
                     </xs:restriction>
                    </xs:simpleType>

<xs:simpleType name="string50">
                     <xs:restriction base="xs:string">
                      <xs:maxLength value="50"/>
                     </xs:restriction>
                    </xs:simpleType>
</xs:schema>
```

XPCML 語法:

XPCML 綱目定義了數個元素標籤，每一個元素標籤均包含屬性標籤。

下表列出您可以在 XPCML 來源檔中宣告並定義的不同元素。第一個直欄中的每一個項目都能鏈結到適當的 XPCML 綱目區段。

XPCML 標籤	說明	相等的 PCML 標籤
doubleParm	定義倍整數參數	資料 (type=float, length=8)
arrayOfDoubleParm	定義作爲倍整數陣列的參數	
floatParm	定義浮點參數	資料 (type=float, length=4)
arrayOfFloatParm	定義作爲浮點陣列的參數	
hexBinaryParm	定義以十六進位來代表的位元組參數	位元組 (約等於，以十六進位代表)
arrayOfHexBinaryParm	定義作爲 hexBinaries 陣列的參數	
intParm	定義整數參數	資料 (type=int, length=4)
arrayOfIntParm	定義作爲整數陣列的參數	
longParm	定義長參數	資料 (type=int, length=8)
arrayOfLongParm	定義作爲長陣列的參數	
packedDecimalParm	定義壓縮十進位參數	資料 (type=packed)
arrayOfPackedDecimalParm	定義作爲壓縮十進位陣列的參數	
parameterList	包含代表程式的所有參數定義之標籤的信號	
program	開始與結束可說明程式呼叫的 XML	程式
shortParm	定義短參數	資料 (類型整數，長度 2)

XPCML 標籤	說明	相等的 PCML 標籤
arrayOfShortParm	定義作爲短陣列的參數	
stringParm	定義字串參數	
arrayOfStringParm	定義作爲字串陣列的參數	
struct	定義一個已命名的結構，您可將它指定爲程式的引數或另一個已命名結構內的欄位。	struct
arrayOfStruct	定義結構陣列	
structParm	代表在 XPCML 文件別處找到的結構標籤之參照，您要將它納入文件中的特定位置	資料 (type=struct)
arrayOfStructParm	定義作爲結構參數陣列的參數	
unsignedIntParm	定義無正負號整數參數	資料 (type=int, length=4, precision=32)
arrayOfUnsignedIntParm	定義作爲無正負號整數陣列的參數	
unsignedShortParm	定義無正負號短參數	資料 (type=int, length=2, precision=16)
arrayOfUnsignedShortParm	定義作爲無正負號短陣列的參數	
xpcml	開始與結束可說明程式呼叫格式的 XPCML 來源檔	
zonedDecimalParm	定義區化的十進位參數	資料 (類型區化)
arrayOfZonedDecimalParm	定義作爲區化十進位陣列的參數	

XPCML 標籤屬性:

XPCML 綱目定義了數個元素標籤，每一個元素標籤均包含屬性標籤。下表列出並說明每一個元素的不同屬性。

有關 XPCML 標籤及其屬性更詳盡特定的資訊，請參閱 XPCML 綱目。

XPCML 標籤	屬性	說明
hexBinaryParm	在最後 2 個欄位填入資料並決定格式	
arrayOfHexBinaryParm		
doubleParm	定義倍整數參數	浮點 (長度 8)
arrayOfDoubleParm	定義作爲倍整數陣列的參數	
floatParm	定義浮點參數	資料 (類型浮點，長度 4)
arrayOfFloatParm	定義作爲浮點陣列的參數	
intParm	定義整數參數	資料 (類型整數，長度 4)
arrayOfIntParm	定義作爲整數陣列的參數	
longParm	定義長參數	資料 (類型整數，長度 8)
arrayOfLongParm	定義作爲長陣列的參數	
packedDecimalParm	定義壓縮十進位參數	資料 (類型壓縮)
arrayOfPackedDecimalParm	定義作爲壓縮十進位陣列的參數	
parameterList	包含代表程式的所有參數定義之標籤的信號	
程式	開始與結束可說明程式呼叫的 XML	
shortParm	定義短參數	資料 (類型整數，長度 2)

XPCML 標籤	屬性	說明
arrayOfShortParm	定義作爲短陣列的參數	
stringParm	定義字串參數	
arrayOfStringParm	定義作爲字串陣列的參數	
struct	定義一個已命名的結構，您可將它指定爲程式的引數或另一個已命名結構內的欄位。	
arrayOfStruct	定義結構陣列	
structParm	代表在 XPCML 文件別處找到的結構標籤之參照，您要將它納入文件中的特定位置	資料 (類型 struct)
arrayOfStructParm	定義作爲結構參數陣列的參數	
unsignedIntParm	定義無正負號整數參數	資料 (類型整數，長度 4，精準度 32)
arrayOfUnsignedIntParm	定義作爲無正負號整數陣列的參數	
unsignedShortParm	定義無正負號短參數	資料 (類型整數，長度 2，精準度 16)
arrayOfUnsignedShortParm	定義作爲無正負號短陣列的參數	
xpcml	開始與結束可說明程式呼叫格式的 XPCML 來源檔	
zonedDecimalParm	定義區化的十進位參數	資料 (類型劃分區域)
arrayOfZonedDecimalParm	定義作爲區化十進位陣列的參數	

使用 XPCML

使用 XPCML 與使用 PCML 類似。下列步驟可作爲您在使用 XPCML 時所需執行之動作的一般概述。

1. 使用 XPCML 來說明程式呼叫的規格
2. 建立 ProgramCallDocument 物件
3. 使用 ProgramCallDocument.callProgram() 來執行程式

儘管與使用 PCML 有相似之處，但使用 XPCML 能爲您帶來強化的功能性：

- 讓剖析器自動驗證參數值
- 指定與傳送程式參數值
- 以 XPCML 來擷取對 iSeries 伺服器的程式呼叫結果
- 將現有的 PCML 文件轉換爲同等的 XPCML 文件
- 延伸 XPCML 綱目，以定義新的簡式或複合元素與屬性

例如，IBM Toolbox for Java 可讓您延伸 XPCML 綱目，以建立新的參數及資料類型。您可以使用 XPCML 的這項功能來壓縮 XPCML 來源檔，這會讓檔案更容易讀取而且程式碼更容易使用。

如需有關使用 XPCML 的詳細資訊，請參閱下列頁面：

將 PCML 轉換至 XPCML
瞭解如何將現有的 PCML 文件轉換爲同等的 XPCML 文件。

使用 XPCML 來呼叫 iSeries 伺服器上的程式
瞭解如何建立 ProgramCallDocument 物件，該物件會使用 XPCML 來呼叫伺服器上的程式。

取得程式呼叫結果作爲 XPCML
瞭解如何擷取對伺服器程式的呼叫結果，作爲 XPCML。

作爲 XPCML 傳入參數值
瞭解如何在 XPCML 中設定程式參數的值並傳入那些值。檢查用來定義常數參數值以及資料陣列的技術。

使用壓縮的 XPCML
瞭解如何透過壓縮 XPCML 文件來使它更容易讀取及使用。 找出如何使用壓縮的 XPCML 建立 ProgramCallDocument 物件，以及如何取得程式呼叫結果作爲壓縮的 XPCML。

在 XPCML 中識別剖析錯誤
瞭解如何識別並記錄有用的警告，以及在剖析 XPCML 文件偶爾會發生的不嚴重剖析錯誤。

將現有的 PCML 轉換為 XPCML:

ProgramCallDocument 類別包含了 transformPCMLToXPCML 方法，它能讓您將現有的 PCML 文件轉換爲同等的 XPCML 文件。

就所有元素及屬性而言，XPCML 具有與您在 PCML 中的定義相提並論的定義。使用 transformPCMLToXPCML()， 就能將元素及屬性的 PCML 表示方式轉換爲同等的 XPCML。

請注意，在某些情況下，同等的 XPCML 屬性與 PCML 中的名稱不同。例如， 在 PCML 中的 usage 屬性，在 XPCML 中卻稱爲 passDirection 屬性。有關相較於 PCML 而使用 XPCML 的詳細資訊，請參閱 XPCML 綱目及語法。

該方法會使用您傳送給它作爲 InputStream 物件的現有 PCML 文件，產生同等的 XPCML 作爲 OutputStream 物件。由於 transformPCMLToXPCML() 是一個靜態方法， 因此您不必先建立 ProgramCallDocument 物件就能呼叫它。

範例：轉換 PCML 文件至 XPCML 文件

下列範例說明如何將 PCML 文件 (稱爲 myPCML.pcml) 轉換至 XPCML 文件 (稱爲 myXPCML.xpcml)。

註: 您必須指定 .xpcml 作爲 XPCML 檔的副檔名。若使用 .xpcml 作爲副檔名，能確保 ProgramCallDocument 類別將該檔案辨識爲 XPCML。如果您不指定副檔名， ProgramCallDocument 就會假設該檔案是 PCML。

PCML 文件 myPCML.pcml

```
<!-- myPCML.pcml -->
<pcml version="4.0">
   <program name="prog1" path="/QSYS.LIB/W95LIB.LIB/PROG1.PGM">
      <data type="char" name="parm1" usage="in" passby="reference"
            minvrm="V5R2M0" ccsid="37" length="10" init="Value 1"/>
   </program>
</pcml>
```

用來將 myPCML.pcml 轉換為 myPCML.xpcml 的 Java 程式碼

```
      try {
  InputStream pcmlStream = new FileInputStream("myPCML.pcml");
  OutputStream xpcmlStream = new FileOutputStream("myXPCML.xpcml");
  ProgramCallDocument.transformPCMLToXPCML(pcmlStream, xpcmlStream);
}
catch (Exception e) {
  System.out.println("error:  - "+e.getMessage());
  e.printStackTrace();
}
```

結果 XPCML 文件 myXPCML.xpcml

```
<?xml version="1.0" encoding="UTF-8"?>
   <!-- myXPCML.xpcml -->
   <xpcml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
        xsi:noNamespaceSchemaLocation="xpcml.xsd" version="4.0">
   <program name="prog1" path="/QSYS.LIB/W95LIB.LIB/PROG1.PGM">
      <parameterList>
         <stringParm name="parm1" passDirection="in" passMode="reference"
                     minvrm="V5R2M0" ccsid="37" length="10">Value 1
         </stringParm>
      </parameterList>
   </program>
</xpcml>
```

有關 transformPCMLToXPCML() 與 ProgramCallDocument 類別的詳細資訊，請參閱下列頁面：

ProgramCallDocument javadoc 資訊

使用 XPCML 來呼叫 iSeries 伺服器上的程式:

當您建立 XPCML 檔案之後，您必須建立 ProgramCallDocument 物件，它能使用 XPCML 規格與資料值來呼叫 iSeries 伺服器上的程式。

透過在 ProgramCallDocument 建構子上傳入 XPCML 的檔名，就能建立 XPCML ProgramCallDocument。以這種方式建立 XPCML ProgramCallDocument，首先會剖析與驗證您的 XPCML 文件，然後會建立 ProgramCallDocument 物件。

若要剖析與驗證 XPCML 文件，請確定您的 CLASSPATH 中包含了執行完整驗證的 XML 剖析器。如需有關執行 XPCML 之基本需求的詳細資訊，請參閱下列頁面：

第 370 頁的『XML 剖析器與 XSLT 處理器』

下列範例顯示，如何針對 XPCML 檔案 myXPCML.xpcml 來建立 ProgramCallDocument 物件。

```
system = new AS400();
// Create a ProgramCallDocument into which to parse the file.
ProgramCallDocument xpcmlDoc =
   new ProgramCallDocument(system, "myXPCML.xpcml");
```

建立 XPCML ProgramCallDocument 與建立 PCML ProgramCallDocument 的唯一差別在於，您傳送給建構子的是 XPCML 文件而不是 PCML 文件。

註: 您必須指定 .xpcml 作爲 XPCML 檔的副檔名。若使用 .xpcml 作爲副檔名，能確保 ProgramCallDocument 類別將該檔案辨識爲 XPCML。如果您不指定副檔名， ProgramCallDocument 就會假設該檔案是 PCML。

使用 XPCML 來呼叫 iSeries 伺服器上的程式

當您建立 ProgramCallDocument 物件之後， 就可以使用 ProgramCallDocument 類別的任何方法來處理 XPCML 文件。例如，透過使用 ProgramCallDocument.callProgram() 來呼叫 iSeries 程式，或者變更 XPCML 輸入參數的值，然後再用適當的 ProgramCallDocument.setValue 方法來呼叫伺服器程式。

下列範例顯示，如何針對 XPCML 檔案 (稱爲 myXPCML.xpcml)，來建立 ProgramCallDocument 物件。當您建立 ProgramCallDocument 物件之後，該範例就會呼叫 XPCML 文件中所指定的程式 (PROG1)。在這種情況下，使用 XPCML 與 PCML 的唯一差別就在於，該範例向 ProgramCallDocument 建構子傳送了 XPCML 檔案。

當您的應用程式讀取並剖析 XPCML 文件之後，XPCML 文件的功能就會像 PCML 文件一樣。這時，XPCML 就能使用 PCML 所使用的任何現有的方法。

```
system = new AS400();

// Create a ProgramCallDocument into which to parse the file.
```

```
ProgramCallDocument xpcmlDoc = new ProgramCallDocument(system, "myXPCML.xpcml");

// Call PROG1
boolean rc = xpcmlDoc.callProgram("PROG1");
```

取得程式呼叫結果作為 XPCML:

呼叫伺服器程式之後，您就可以使用 ProgramCallDocument.getValue 方法，以擷取代表程式參數值的 Java 物件。

此外，下列 generateXPCML 方法還能讓 ProgramCallDocument 傳回程式呼叫結果作爲 XPCML：

- generateXPCML(「字串」fileName)：針對您要用來建構 ProgramCallDocument 物件的整個 XPCML 來源檔，產生 XPCML 的結果。 將 XPCML 儲存在具有特定檔名的檔案中。
- generateXPCML(「字串」pgmName，「字串」fileName)：僅針對特定的程式及其參數產生 XPCML 的結果。將 XPCML 儲存在具有特定檔名的檔案中。
- generateXPCML(java.io.OutputStream outputStream)：針對整個 XPCML 來源檔產生 XPCML 的結果。將 XPCML 儲存在特定的 OutputStream 物件中。
- generateXPCML(「字串」pgmName， java.io.OutputStream outputStream)：僅針對特定的程式及其參數產生 XPCML 的結果。將 XPCML 儲存在特定的 OutputStream 物件中。

如需有關 ProgramCallDocument 類別的詳細資訊，請參閱下列頁面：

ProgramCallDocument javadoc 資訊

下列範例說明如何建構 XPCML ProgramCallDocument、呼叫 iSeries 程式，以及以 XPCML 的形式擷取程式呼叫結果。

第 688 頁的『範例：擷取程式呼叫結果作爲 XPCML』

作為 XPCML 傳入參數值:

您可以在 XPCML 來源檔中設定程式參數值，並傳入該參數值作爲 XPCML。

當 ProgramCallDocument 物件讀取並剖析 XPCML 文件時，它就會針對 XPCML 中所指定的每一個參數自動呼叫適當的 setValue 方法。

使用 XPCML 來傳入參數值，可讓您不必再費神撰寫 Java 程式碼來設定複雜的結構與陣列值。

下列範例將說明建構陣列並傳入參數值作爲 XPCML 的不同方式。

第 691 頁的『範例：傳入參數值作爲 XPCML』

第 692 頁的『範例：傳入參數值陣列作爲 XPCML』

使用壓縮的 XPCML:

由於 XPCML 是可延伸的，因此您可以定義新的參數類型，來延伸那些由 XPCML 綱目指定的參數類型。壓縮 XPCML 能延伸 XPCML 綱目，建立新的資料類型， 以簡化並增進 XPCML 文件的可靠性及實用性。

下列討論假設您已瞭解 XPCML 綱目。如需有關 XPCML 綱目的詳細資訊，請參閱下列頁面：

第 373 頁的『XPCML 綱目及語法』

若要壓縮現有的 XPCML 來源檔，您可以使用 ProgramCallDocument.condenseXPCML 方法，它會產生下列項目：

- 一個延伸的綱目，其中包含現有 XPCML 來源當中每一個參數的新類型定義
- 新的 XPCML 來源檔，它會使用延伸的綱目中所提供的類型定義

有關壓縮 XPCML 的詳細資訊，請參閱下列頁面：

第 395 頁的『使用壓縮的 XPCML』

第 697 頁的『範例：使用壓縮的 XPCML 來建立 ProgramCallDocument 物件』

第 697 頁的『範例：取得程式呼叫結果作爲壓縮的 XPCML』

壓縮現有的 XPCML 文件:

壓縮現有的 XPCML 文件能使 XPCML 來源檔更加易於讀取與使用。若要建立壓縮的 XPCML， 請使用 ProgramCallDocument.condenseXPCML 方法。

若要呼叫 condenseXPCML()，請爲該方法提供下列參數：

- 代表現有 XPCML 的輸入串流
- 代表壓縮 XPCML 的輸出串流
- 代表新的延伸 XPCML 的輸出串流
- 具有適當格式的新綱目名稱 (如 mySchema.xsd)

如需有關 condenseXPCML() 與 ProgramCallDocument 類別的詳細資訊，請參閱下列頁面：

ProgramCallDocument javadoc 資訊

ProgramCallDocument.condenseXPCML() 是一個靜態方法，這表示您不必案例化 ProgramCallDocument 物件就能呼叫該方法。

範例

下列範例說明壓縮現有 XPCML 文件的方式。

第一個範例比較簡單，其中包含了原始 XPCML 來源、已壓縮的結果 XPCML 以及延伸的綱目。第二個範例較長而複雜，因此它包含了呼叫 condenseXPCML() 的 Java 程式碼，以及延伸的綱目中少數幾個新產生的類型定義：

第 694 頁的『範例：壓縮現有的 XPCML 文件』

第 694 頁的『範例：壓縮現有的 XPCML 文件』

在 XPCML 中識別剖析錯誤:

在驗證 XPCML 綱目文件時，執行完整剖析的 XML Parser 可能會產生警告、 不嚴重的解析錯誤以及嚴重的解析錯誤。

警告與不嚴重的解析錯誤不會讓剖析失敗。您可以檢查警告與不嚴重的解析錯誤，以判定 XPCML 來源檔中的問題。嚴重的解析錯誤會導致剖析終止並傳回異常。

若要在剖析 XPCML 文件時顯示警告與不嚴重的剖析錯誤，請開啓應用程式中的追蹤功能，並將追蹤種類設到 PCML。

範例

執行完整驗證的 XML 剖析器，會針對不含數值的數字參數產生錯誤訊息。下列範例會顯示一個 XPCML 來源範例以及所產生的不嚴重解析錯誤：

XPCML 來源

```
<program name="prog1"/>
   <parameterList>
      <intParm name="parm1"/>
   </parameterList>
</program>
```

產生的錯誤

```
Tue Mar 25 15:21:44 CST 2003  [Error]: cvc-complex-type.2.2: Element
'intParm' must have no element [children], and the value must be valid.
```

若要避免記錄這類錯誤，請在 intParm 元素中加入 nil=true 屬性。nil=true 屬性會示意剖析器，是您刻意要將該元素留空的。下面就是先前的 XPCML 來源加入了 nil=true 屬性：

```
<program name="prog1"/>
   <parameterList>
      <intParm xsi:nil="true" name="parm1"/>
   </parameterList>
</program>
```

常見問題 (FAQ)

IBM Toolbox for Java 常見問題集 (FAQ) 針對 IBM Toolbox for Java 效能的最佳化、執行疑難排解、使用 JDBC 及其他動作有關的問題，提供了適當的回答。

- IBM Toolbox for Java FAQ  ：您可在此找到許多問題的回答，包括提升效能、使用 i5/OS、執行疑難排解等。
- IBM Toolbox for Java JDBC FAQ  ：您可在此找到 JDBC 與 IBM Toolbox for Java 搭配使用的相關問題回答

程式設計秘訣

本節將提供各種秘訣，協助您使用 IBM Toolbox for Java。

關閉 Java 程式

若要確定您的程式是否已正確關閉，請在結束 Java 程式之前，發出 System.exit(0) 作爲最後一個指示。

註: 請避免在 Servlet 中使用 System.exit(0)，因爲這樣做，會關閉整個 Java 虛擬機器。

IBM Toolbox for Java 會透過使用者緒連接到伺服器。因此，當無法發出 System.exit(0) 時，可能會使得您的 Java 程式無法正確關閉。

使用 System.exit(0) 並非必要，而是預防措施。您會有必須使用這個指令來結束 Java 程式的時候，而在不必要時使用 System.exit(0) 也不會有什麼問題。

伺服器物件的整合檔案系統路徑名稱

您的 Java 程式必須使用整合檔案系統名稱來參照伺服器物件，例如程式、程式庫、指令或排存檔。整合檔案系統名稱是伺服器物件的名稱，因爲它可在 iSeries 伺服器上的整合檔案系統的檔案庫檔案系統中存取。

路徑名稱可以由下列元件所組成：

路徑名稱元件	說明
檔案庫	檔案庫常駐的檔案庫。檔案庫是整合檔案系統路徑名稱的**必要**部分。檔案庫名稱必須是 10 個或更少的字元，且後面必須接著 **.lib**。
物件	整合檔案系統路徑名稱代表的物件的名稱。物件是整合檔案系統路徑名稱的**必要**部分。物件名稱必須是 10 個或更少的字元，且其後須接著 **.type**， 其中 **type** 是物件的類型。您可透過控制語言 (CL) 指令 (如「使用物件 (WRKOBJ)」) 上的 OBJTYPE 參數的提示，來找到想要的類型。
類型	物件的類型。當指定**物件**時，亦必須指定物件的類型。(請參閱上面的**物件**。)類型名稱必須是 6 個或更少的字元。
成員	此整合檔案系統路徑名稱代表的成員的名稱。成員是整合檔案系統路徑名稱的**選用**部分。只有在**物件類型**是 **FILE** 時，才能指定它。成員名稱必須是 10 個或更少的字元，且其後須接著 **.mbr**。

當決定及指定整合檔案系統名稱時，請遵循這些條件：

- 正斜線 (/) 爲路徑分隔字元。
- 名爲 QSYS.LIB 的根目錄含有伺服器檔案庫結構。
- 常駐在伺服器檔案庫 QSYS 中的物件具有下列格式：

  ```
  /QSYS.LIB/object.type
  ```

- 常駐在其它檔案庫中的物件具有下列格式：

  ```
  /QSYS.LIB/library.LIB/object.type
  ```

- 物件類型副檔名即是針對該類型的物件所使用的伺服器縮寫。

若要察看這些類型的清單，請輸入一個具有物件類型作爲參數的 CL 指令， 然後按 **F4** (提示) 取得類型的提示。例如，「使用物件 (WRKOBJ)」指令即具有物件類型參數。

下表是一些常用物件類型的清單及每一種類型的縮寫：

物件類型	縮寫
指令	.CMD
資料佇列	.DTAQ
檔案	.FILE
字型資源	.FNTRSC
格式定義	.FORMDF
檔案庫	.LIB
成員	.MBR
套印格式	.OVL
頁定義	.PAGDFN

物件類型	縮寫
頁區段	.PAGSET
程式	.PGM
輸出佇列	.OUTQ
排存檔	.SPLF

您可以使用下列說明，決定如何指定整合檔案系統路徑名稱：

整合檔案系統名稱	說明
/QSYS.LIB/MY_LIB.LIB/MY_PROG.PGM	伺服器上檔案庫 MY_LIB 中的程式 MY_PROG
/QSYS.LIB/MY_LIB.LIB/MY_QUEUE.DTAQ	伺服器上檔案庫 MY_LIB 中的資料佇列 MY_QUEUE
/QSYS.LIB/YEAR1998.LIB/MONTH.FILE/JULY.MBR	伺服器上檔案庫 YEAR1998 的檔案 MONTH 中的成員 JULY

整合檔案系統特殊值

有多種 IBM Toolbox for Java 類別可辨識整合檔案系統路徑名稱中的特殊值。這些特殊值的傳統格式 (例如用於 iSeries 指令行) 會以星號 (***ALL**) 開頭。不過，在使用 IBM Toolbox for Java 類別的 Java 程式中，這些特殊值的格式會以百分比符號 (**%ALL%**) 作爲開頭及結尾。

註: 在整合檔案系統中，星號是萬用字元。

下表列出 IBM Toolbox for Java 類別可辨識特定路徑名稱元件的哪些特殊值。該表格也會列出這些特殊值的傳統格式與 IBM Toolbox for Java 類別中所使用的格式有何不同。

路徑名稱元件	傳統格式	IBM Toolbox for Java 格式
檔案庫名稱	*ALL	%ALL%
	*ALLUSR	%ALLUSR%
	*CURLIB	%CURLIB%
	*LIBL	%LIBL%
	*USRLIBL	%USRLIBL%
物件名稱	*ALL	%ALL%
成員名稱	*ALL	%ALL%
	*FILE	%FILE%
	*FIRST	%FIRST%
	*LAST	%LAST%

請參閱 QSYSObjectPathName 類別，取得關於建置與剖析整合檔案系統名稱的資訊。

有關整合檔案系統概念的詳細資訊，請參閱整合檔案系統概念。

管理連線

能建立、啓動及結束與伺服器的連接是很重要的。以下討論說明管理與伺服器連接的主要概念， 並提供一些程式碼範例。

若要與 iSeries 伺服器連接，您的 Java 程式必須建立 AS400 物件。AS400 物件對每一種 iSeries 伺服器類型，最多含有一個 socket 連接。一個服務程式會對應一個伺服器上的工作，且爲伺服器中資料的介面。

註: 當您在建立 Enterprise JavaBeans (EJB) 時，請遵守連線期間不允許使用緒的 EJB 規格。雖然關閉 IBM Toolbox for Java 緒支援可能會使應用程式的執行速度降低，但仍應遵守 EJB 規格。

在 iSeries 上，每一個與各伺服器的連接均有自己的工作。分別使用不同的伺服器來支援下列每一項：

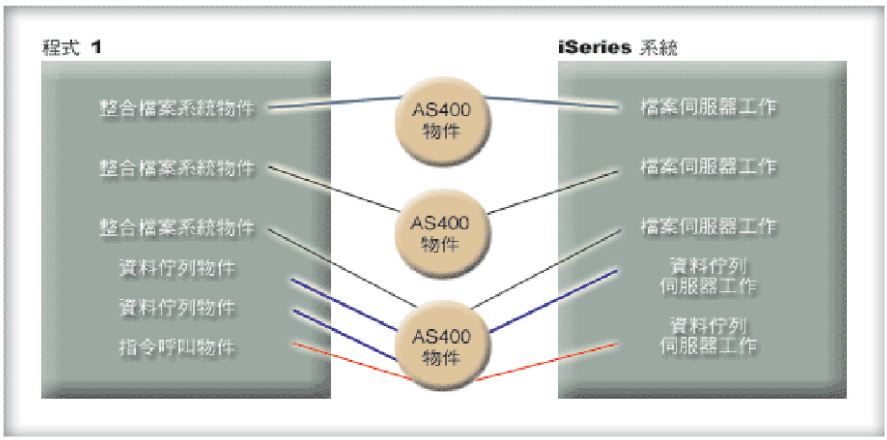
- JDBC
- 程式呼叫與指令呼叫
- 整合檔案系統
- 列印
- 資料佇列
- 記錄層次存取

註:

- 如果應用程式不嘗試同時執行需要網路列印伺服器的兩個工作，則 print 類別會對每一個 AS400 物件使用 一個 socket 連接。
- 必要時，列印類別將會對網路列印伺服器建立額外的 socket 連接。如果 5 分鐘內未使用額外互動，將切斷它們。

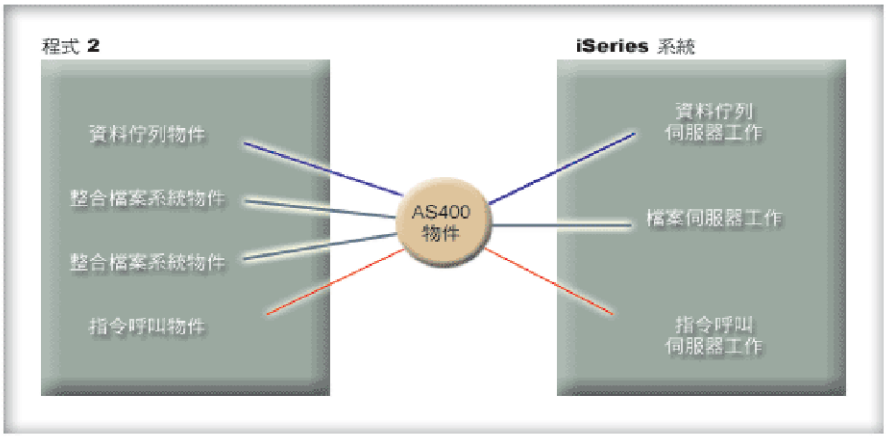
Java 程式可控制與 iSeries 的連接數。爲了達到最佳的通訊效能，Java 程式可以爲同一部伺服器建立多個 AS400 物件，如圖 1 所示。如此將會針對 iSeries 伺服器建立多個 Socket 連線。

圖 1：為同一部 iSeries 伺服器建立多個 AS400 物件及 Socket 連線的 Java 程式



若要保留伺服器資源，請只建立一個 AS400 物件，如圖 2 所示。此方法會降低連線數量，因而減少伺服器上的資源使用量。

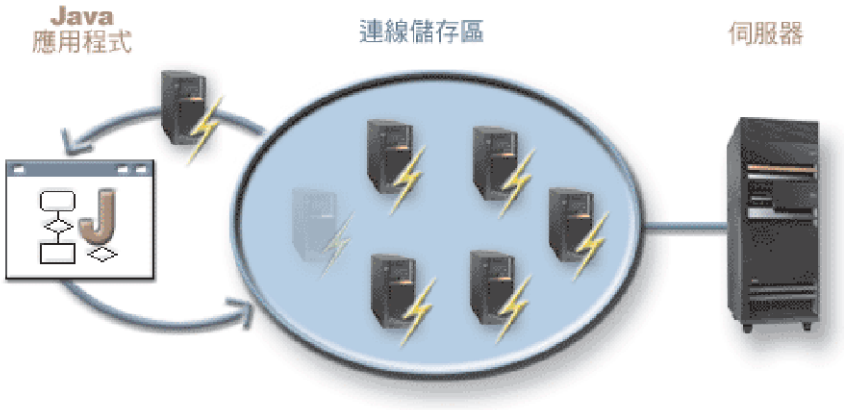
圖 2：為同一部 iSeries 伺服器建立單一 AS400 物件及 Socket 連線的 Java 程式



註: 雖然建立多個連線會增加伺服器上的資源使用量，但建立多個連線還是有好處的。具有多個連線可讓您的 Java 程式以平行方式處理工作，因而使產能 (每秒異動數) 增加並加速應用程式的執行。

您也可選擇使用連線儲存區來管理連線，如圖 3 所示。此方法會重覆使用先前建立給使用者的連線，因而減少了與 iSeries 連接所要花費的時間。

圖 3：由 AS400ConnectionPool 取得 iSeries 伺服器連線的 Java 程式



下列範例說明如何建立及使用 AS400 物件：

範例 1： 在下列範例中， 將建立兩個 CommandCall 物件，傳送指令給同一部伺服器。因爲 CommandCall 物件使用相同的 AS400 物件，所以僅會建立一個伺服器連線。

```
        // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

        // Create two command call objects that use
        // the same AS400 object.
CommandCall cmd1 = new CommandCall(sys,"myCommand1");
CommandCall cmd2 = new CommandCall(sys,"myCommand2");

        // Run the commands.  A connection is made when the
        // first command is run.  Since they use the same
        // AS400 object the second command object will use
        // the connection established by the first command.
cmd1.run();
cmd2.run();
```

範例 2： 在下列範例中，將建立兩個 CommandCall 物件，以傳送指令給同一部 iSeries 伺服器。因爲 CommandCall 物件使用不同的 AS400 物件，所以會建立兩個伺服器連線。

```
        // Create two AS400 objects to the same server.
AS400 sys1 = new AS400("mySystem.myCompany.com");
AS400 sys2 = new AS400("mySystem.myCompany.com");

        // Create two command call objects.  They use
        // different AS400 objects.
CommandCall cmd1 = new CommandCall(sys1,"myCommand1");
CommandCall cmd2 = new CommandCall(sys2,"myCommand2");

        // Run the commands.  A connection is made when the
        // first command is run.  Since the second command
        // object uses a different AS400 object, a second
        // connection is made when the second command is run.
cmd1.run();
cmd2.run();
```

範例 3： 在下列範例中，將使用同一個 AS400 物件，建立 CommandCall 物件與 IFSFileInputStream 物件。因爲在 iSeries 伺服器上，CommandCall 物件與 IFSFileInput Stream 物件使用不同的服務程式，因此將建立兩個連線。

```
         // Create an AS400 object.
AS400 newConn1 = new AS400("mySystem.myCompany.com");

        // Create a command call object.
CommandCall cmd = new CommandCall(newConn1,"myCommand1");

        // Create the file object.  Creating it causes the
        // AS400 object to connect to the file service.
IFSFileInputStream file = new IFSFileInputStream(newConn1,"/myfile");

        // Run the command.  A connection is made to the
        // command service when the command is run.
cmd.run();
```

範例 4： 在下列範例中，將使用 AS400ConnectionPool 來取得 iSeries 連線。 本範例 (如同以上的範例 3) 不會指定服務程式，因此會於執行指令時進行與指令服務程式的連線。

```
        // Create an AS400ConnectionPool.
AS400ConnectionPool testPool1 = new AS400ConnectionPool();
        // Create a connection.
AS400 newConn1 = testPool1.getConnection("myAS400", "myUserID", "myPassword");
        // Create a command call object that uses the AS400 object.
CommandCall cmd = new CommandCall(newConn1,"myCommand1");
        // Run the command.  A connection is made to the
        // command service when the command is run.
cmd.run();
        // Return connection to pool.
testPool1.returnConnectionToPool(newConn1);
```

範例 5：下列範例中，在要求儲存區的 連線時，會使用 AS400ConnectionPool 連接到特定的服務程式。如此會在執行指令時，減少與服務程式連接所需的時間 (請參閱以上的範例 4)。如果連線傳回儲存區，則下一個取得連線的呼叫會傳回相同的連線物件。這表示毋需額外的連接時間，不論是在建立時或使用時都一樣。

```
        // Create an AS400ConnectionPool.
AS400ConnectionPool testPool1 = new AS400ConnectionPool();
        // Create a connection to the AS400.COMMAND service. (Use the service number constants
        // defined in the AS400 class (FILE, PRINT, COMMAND, DATAQUEUE, and so on.))
AS400 newConn1 = testPool1.getConnection("myAS400", "myUserID", "myPassword", AS400.COMMAND);
        // Create a command call object that uses the AS400 object.
CommandCall cmd = new CommandCall(newConn1,"myCommand1");
        // Run the command.  A connection has already been made
        // to the command service.
```

```
cmd.run();
        // Return connection to pool.
testPool1.returnConnectionToPool(newConn1);
        // Get another connection to command service.  In this case, it will return the same
        // connection as above, meaning no extra connection time will be needed either now or
        // when the command service is used.
AS400 newConn2 = testPool1.getConnection("myAS400", "myUserID", "myPassword", AS400.COMMAND);
```

啓動與結束連線

Java 程式可控制連線的啓動與結束時間。根據預設值，當伺服器需要資訊時，即會啓動一個連線。您可藉由在 AS400 物件中呼叫 connectService() 方法，預先連接至伺服器，來控制進行連線的正確時機。

您可使用一個 AS400ConnectionPool， 在不呼叫 connectService() 方法的情況下，建立與服務程式先前連接的連線，如以上的範例 5 所述。

下列範例說明與 iSeries 連接及切斷連接的 Java 程式。

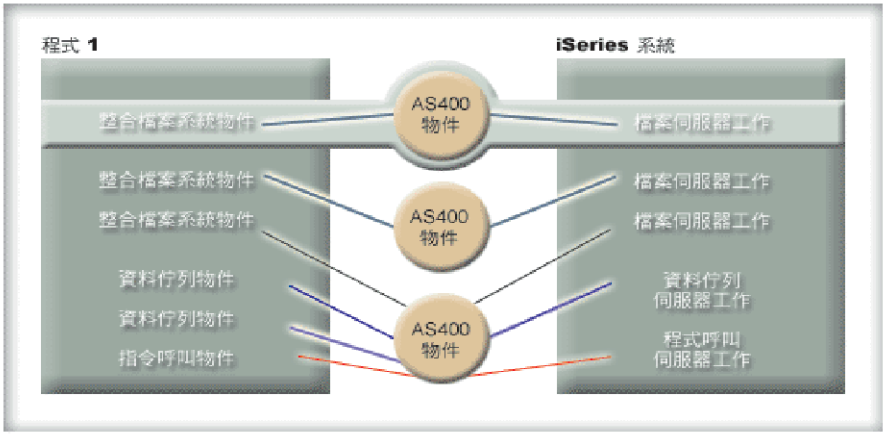
範例 1：此範例將說明如何預先連接至 iSeries：

```
         // Create an AS400 object.
AS400 system1 = new AS400("mySystem.myCompany.com");

        // Connect to the command service.  Do it now
        // instead of when data is first sent to the
        // command service.  This is optional since the
        // AS400 object will connect when necessary.
system1.connectService(AS400.COMMAND);
```

範例 2：啓動連線之後，Java 程式將負責切斷連線，其方式可能是隱含地透過 AS400 物件，或是明確地透過 Java 程式。Java 程式會藉著在 AS400 物件中呼叫 disconnectService() 方法來切斷連線。若要增進效能，則只有在透過服務結束程式時，Java 程式才能切斷連線。如果 Java 程式在服務完成執行之前已斷線，則當需要從服務取得資料時，AS400 物件將重新連線 (如果可能重新連線的話)。

圖 4 說明切斷第一個整合檔案系統物件連接會如何僅結束 AS400 物件連線的單一案例，而非所有的整合檔案系統物件連線。

圖 4：對 AS400 物件的實例使用自己的服務程式之單一物件將被切斷連線

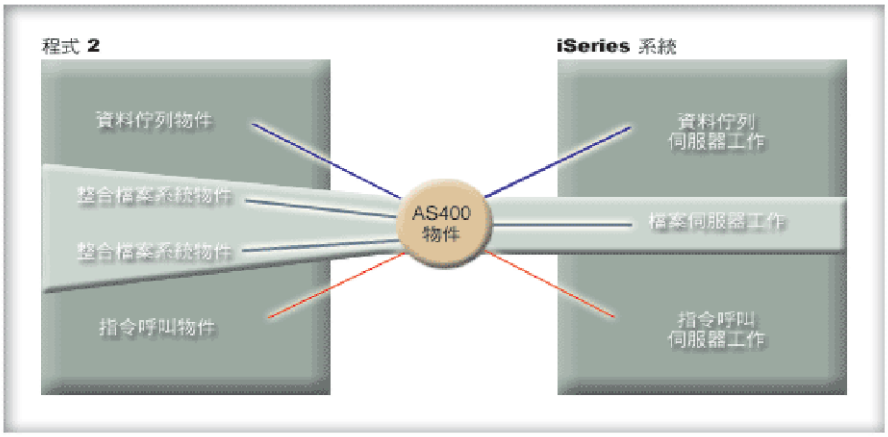


本範例說明 Java 程式如何切斷連線：

```
        // Create an AS400 object.
AS400 system1 = new AS400("mySystem.myCompany.com");

                                // ... use command call to send several commands
                                // to the server.  Since connectService() was not
                    // called, the AS400 object automatically
                                // connects when the first command is run.

                                // All done sending commands so disconnect the
                                // connection.
        system1.disconnectService(AS400.COMMAND);
```

範例 3：使用同一個服務並共用同一個 AS400 物件的多重物件將共用連線。 切斷連線將結束對 AS400 物件之各案例使用同一個服務程式之所有物件的連線，如圖 5 中所示。

圖 5： 對 AS400 物件的實例使用同一個服務程式的所有物件將被切斷連線



例如，有兩個 CommandCall 物件使用同一個 AS400 物件。 當呼叫 disconnectService() 時， 將同時結束這兩個 CommandCall 物件的連接。當呼叫第二個 CommandCall 物件的 run() 方法時，AS400 物件必須重新連接至服務程式：

```
            // Create an AS400 object.
     AS400 sys = new AS400("mySystem.myCompany.com");

                                // Create two command call objects.
CommandCall cmd1 = new CommandCall(sys,"myCommand1");
CommandCall cmd2 = new CommandCall(sys,"myCommand2");

                                // Run the first command
        cmd1.run();

                                // Disconnect from the command service.
        sys.disconnectService(AS400.COMMAND);

                                // Run the second command.  The AS400 object
                    // must reconnect to the server.
        cmd2.run();

                                // Disconnect from the command service.
This
                                // is the correct place to disconnect.
        sys.disconnectService(AS400.COMMAND);
```

範例 4：並非所有的 IBM Toolbox for Java 類別均會自動重新連線。整合檔案系統類別中，有些方法呼叫不會重新連線，因爲檔案可能已有所變更。當檔案被切斷連接時，某些其它處理可能已刪除了檔案，或已變更了

它的內容。在下列範例中，有兩個檔案物件使用同一個 AS400 物件。 當呼叫 disconnectService() 時，將同時結束這兩個檔案物件的連接。 第二個 IFSFileInputStream 物件的 read() 將失敗，因爲它不再具有與伺服器的連線。

```
             // Create an AS400 object.
      AS400 sys = new AS400("mySystem.myCompany.com");

                                    // Create two file objects.  A connection to the
                         // server is created when the first object is
                                    // created.  The second object uses the connection
                         // created by the first object.
     IFSFileInputStream file1 = new IFSFileInputStream(sys,"/file1");
     IFSFileInputStream file2 = new IFSFileInputStream(sys,"/file2");

                                    // Read from the first file, then close it.
          int i1 = file1.read();
          file1.close();

                                    // Disconnect from the file service.
          sys.disconnectService(AS400.FILE);

             // Attempt to read from the second file.  This
                                    // fails because the connection to the file service
                                    // no longer exists.  The program must either
                         // disconnect later or have the second file use a
                                    // different AS400 object (which causes it to
                                    // have its own connection).
          int i2 = file2.read();

                                    // Close the second file.
          file2.close();

                                    // Disconnect from the file service.
This
                                    // is the correct place to disconnect.
          sys.disconnectService(AS400.FILE);
```

圖 1 的詳細說明：對同一個 iSeries 伺服器建立多個 AS400 物件與 Socket 連線的 Java 程式 (rzahh549.gif)

位於 IBM Toolbox for Java：管理連線

本圖說明 Java 程式如何建立多重 AS400 物件，並讓每個物件對同一個 iSeries 伺服器都能具有個別的 Socket 連線。建立多重連線可增加系統中所用的資源。

說明

此圖由下列項目所構成：

- 左邊的矩形代表 Java 程式
- 右邊的矩形代表 iSeries 伺服器
- 矩形之間三個垂直排列的小圓圈，代表 Java 程式所建立的 AS400 物件，連接至 iSeries 伺服器

Java 程式 (左邊的矩形) 包含使用 AS400 物件的數種物件 (三個小圓圈)。這些程式物件上有線連接至 AS400 物件。每一個程式物件都只需要使用一個 AS400 物件。不過，在各種不同類型的程式物件中，有幾項可使用單一 AS400 物件來連接 iSeries。

AS400 物件上有線連接至 iSeries 伺服器所具備的服務 (右邊的矩形)。這些服務適當地鏡映程式物件。以下清單說明多個程式物件、三個 AS400 物件及 iSeries 伺服器服務彼此之間如何連接：

- 一個整合檔案系統程式物件連接第一個 AS400 物件，這個 AS400 物件連接檔案伺服器工作

- 另一個整合檔案系統程式物件連接第二個 AS400 物件，這個 AS400 物件連接第二個檔案伺服器工作
- 第三個 AS400 物件連接著多重程式物件：第三個整合檔案系統物件、兩個資料佇列物件和一個指令呼叫物件。接著， AS400 物件會連接 iSeries 伺服器中的下列服務：第三個檔案伺服器工作 (針對整合檔案系統物件)、資料佇列伺服器工作 (針對資料佇列物件) 和指令呼叫伺服器工作 (針對指令呼叫物件)。

圖 2 的詳細說明：對同一部 iSeries 伺服器建立單一 AS400 物件與 Socket 連線的 Java 程式 (rzahh552.gif)

位於 IBM Toolbox for Java：管理連線

本圖說明安排多重程式物件共用單一 AS400 物件可減少系統上使用的連線數目和資源量。

說明

此圖由下列項目所構成：

- 左邊的矩形代表 Java 程式
- 右邊的矩形代表 iSeries 伺服器
- 矩形之間的小圓圈代表 Java 程式所建立的單一 AS400 物件，連接至 iSeries 伺服器

Java 程式 (左邊的矩形) 包含使用 AS400 物件的數種物件。從這些程式物件有線連接著 AS400 物件 (小圓圈)。

AS400 物件上有線連接至 iSeries 伺服器所具備的服務 (右邊的矩形)。每個服務各代表通往伺服器的不同連線。這些服務適當地鏡映程式物件：

- 有一個資料佇列程式物件使用 AS400 物件，它連接資料佇列伺服器工作
- 有兩個整合式的檔案系統程式物件使用 AS400 物件，它們連接單一的檔案伺服器工作
- 有一個指令呼叫工作物件使用 AS400 物件，它連接指令呼叫伺服器工作

所有這些程式物件全都使用同一個 AS400 物件，此物件可爲伺服器上執行的每個工作各建立單獨的連線。本案例中，因爲整合式檔案系統物件使用同一個 AS400 物件，所以也在伺服器上共用通往工作的連線。

圖 3 的詳細說明：Java 程式從 AS400ConnectionPool 取得與 iSeries 伺服器的連線 (rzahh506.gif)

位於 IBM Toolbox for Java：管理連線

本圖說明如何使用連線儲存區來管理您與 iSeries 伺服器的連線。

說明

此圖由下列項目所構成：

- 左邊的矩形影像代表 Java 應用程式
- 右邊的 iSeries 伺服器圖片代表 Java 應用程式要連接的伺服器
- 左右兩個影像之間的橢圓形代表 AS400ConnectionPool 物件

AS400ConnectionPool (橢圓) 內部有多重連線。每個連線都是一個 AS400 物件，以具有閃電的 iSeries 伺服器小型影像來表示：

- 循環箭頭將 Java 程式 (左邊的矩形) 連接到連線儲存區 (橢圓)。
- 單線將 iSeries 伺服器 (右邊的影像) 連接到連線儲存區 (橢圓)。

循環箭頭的上半部從 AS400ConnectionPool (橢圓) 指向 Java 應用程式 (左邊的矩形影像)。箭頭上有來自儲存區的一條連線，此外連線儲存區有一個連線的影像以淡色調顯示。這代表 Java 應用程式所要求並使用之來自連線儲存區的連線。

循環箭頭的下半部從 Java 應用程式指向連線儲存區。這代表將連線傳回連線儲存區的 Java 應用程式。

將 iSeries 伺服器 (右邊的影像) 連接到 AS400ConnectionPool (橢圓) 的單線代表伺服器的 Socket 連線。

圖 4 的長說明：對 AS400 物件的實例使用自己的服務程式之單一物件將被切斷連接 (rzahh550.gif)

位於 IBM Toolbox for Java：管理連線

本圖說明如何切斷單一 AS400 物件的連線而不影響其它 AS400 物件的連線。

說明

本圖是由圖 1 的相同元素所組成：

- 左邊的矩形代表 Java 程式
- 右邊的矩形代表 iSeries 伺服器
- 矩形之間三個垂直排列的小圓圈，代表 Java 程式所建立的 AS400 物件，連接至 iSeries 伺服器

Java 程式 (左邊的矩形) 包含使用 AS400 物件的數種物件 (三個垂直排列的小圓圈)。這些程式物件上有線連接至 AS400 物件。每一個程式物件都只需要使用一個 AS400 物件。不過，在各種不同類型的程式物件中，有幾項可使用單一 AS400 物件來連接 iSeries。

AS400 物件上有線連接至 iSeries 伺服器所具備的服務 (右邊的矩形)。這些服務適當地鏡映程式物件。如需不同程式物件、AS400 物件及 iSeries 伺服器服務的特定說明，請參閱圖 1 的說明。

單一 AS400 物件的連線以高亮度顯示，說明如何切斷這條連線而不影響其它 AS400 物件的連線。該連線由整合檔案系統物件、連線所使用的 AS400 物件、相關的 iSeries 伺服器服務 (檔案伺服器的工作) 以及連接上述物件與服務的線所組成。至於其它的程式物件、線路、AS400 物件或服務則全都不以高亮度顯示。

圖 5 的長說明：對 AS400 物件的例項使用同一個服務的所有物件都會被切斷連接 (rzahh551.gif)

位於 IBM Toolbox for Java：管理連線

本圖說明當切斷單一 AS400 物件的服務時，將如何連帶使得針對 AS400 物件的那個實例共用同一個服務的所有物件都被切斷。

說明

本圖是由圖 2 的相同元素所組成：

- 左邊的矩形代表 Java 程式
- 右邊的矩形代表 iSeries 伺服器
- 矩形之間的小圓圈代表 Java 程式所建立的單一 AS400 物件，連接至 iSeries 伺服器

Java 程式 (左邊的矩形) 包含使用 AS400 物件的數種物件 (小圓圈)。從這些程式物件有線連接著 AS400 物件。

AS400 物件上有線連接至 iSeries 伺服器所具備的服務 (右邊的矩形)。這些服務適當地鏡映程式物件。如需不同程式物件、AS400 物件及 iSeries 伺服器服務的特定說明，請參閱圖 2 的說明。

有兩個整合檔案系統程式物件使用 AS400 物件，連接至 iSeries 伺服器上的檔案伺服器工作。這兩個程式物件、相關的服務和相連接的線路以高亮度顯示。切斷服務雖然只影響到高亮度顯示的元素，但結果也使得這兩個程式物件都遭到切斷。至於其它的程式物件、服務、連線或 AS400 物件本身則全都不受影響。

i5/OS Java 虛擬機器

IBM Toolbox for Java 類別於 IBM Developer Kit for Java (i5/OS) Java 虛擬機器 (JVM) 上執行。

事實上，這些類別可在任何支援 Java 2 Software Development Kit (J2SDK) 規格的平台上執行。

當您在 i5/OS JVM 上執行 IBM Toolbox for Java 類別時，請執行下列動作：

- 選擇在 i5/OS JVM 中執行時，是否要使用 i5/OS JVM 或 IBM Toolbox for Java 類別來存取 iSeries 伺服器資源。
- 移出 i5/OS JVM 上的執行 IBM Toolbox for Java 類別。
- 讀取 i5/OS JVM 中有關設定系統名稱、使用者 ID 及密碼的資訊。

如需 iSeries 伺服器支援不同 Java 平台的相關資訊，請參閱支援多個 JDK。

比較 i5/OS Java 虛擬機器與 IBM Toolbox for Java 類別

當 Java 程式在 IBM Developer Kit for Java (i5/OS) Java 虛擬機器 (JVM) 中執行時，您至少可使用兩種方法來存取 iSeries 伺服器資源。

您可以使用下列任一個介面：

- 內建在 Java 中的機能
- IBM Toolbox for Java 類別

當決定將使用哪一個介面時，請考慮下列因素：

- **位置** - 在決定要使用哪一個介面時，程式執行之處是最重要的因素。程式將：
  - 僅在用戶端上執行嗎？
  - 僅在伺服器上執行嗎？
  - 同時執行於用戶端與伺服器上，但在這兩個情形下，資源是 iSeries 伺服器資源嗎？
  - 在某一個 i5/OS JVM 中執行，並存取另一部 iSeries 伺服器的資源嗎？
  - 在不同種類的伺服器上執行嗎？

  如果程式同時執行於用戶端與伺服器 (包括作爲第二部 iSeries 伺服器之用戶端的 iSeries 伺服器) 上，並僅存取 iSeries 伺服器資源，則使用 IBM Toolbox for Java 介面可能是最好的方式。

  如果程式必須存取多種類型伺服器中的資料，使用 Java 原生介面可能是最好的方式。
- **一致性 / 可攜性** - 在 iSeries 伺服器上執行 IBM Toolbox for Java 類別的能力表示可以將相同的介面用於用戶端程式及伺服器程式。對用戶端程式與伺服器程式，您僅要學習一個介面時，您將更有生產力。

  不過，將資料寫入 IBM Toolbox for Java 介面將使您的程式較不具**伺服器**可攜性。

  如果程式必須執行於 iSeries 伺服器及其他伺服器上，您可能會發現使用已在 Java 中建置的機能可能更好用。

- **複雜性** - IBM Toolbox for Java 介面的建置是爲了方便存取 iSeries 伺服器資源。通常，使用 IBM Toolbox for Java 介面的另一個唯一選擇方案爲：撰寫一個程式，透過「Java 原生介面 (JNI)」存取資源並與該程式通訊。

  您必須決定下列何者較具重要性：使 Java 更中立，並撰寫一個程式來存取資源，或是使用較不具可攜性的 IBM Toolbox for Java 介面。
- **函數** - IBM Toolbox for Java 介面通常比 Java 介面提供更多的函數。例如，IBM Toolbox for Java 授權程式之 IFSFileOutputStream 類別的函數比 java.io 的 FileOutputStream 類別爲多。不過，使用 IFSFileOutputStream 會使您的程式更適用於 iSeries 伺服器。使用 IBM Toolbox for Java 類別，您會失去**伺服器**的可攜性。

  您必須決定可攜性較重要，還是您想要利用其它的函數。
- **資源** - 執行於 i5/OS JVM 中時，許多 IBM Toolbox for Java 類別仍會透過主電腦伺服器提出要求。因此，第二個工作 (伺服器工作) 將實施存取資源的要求。

  此要求所使用的資源可能會比在 Java 程式之工作下執行的 Java 原生介面所使用的資源要多。
- **iSeries 伺服器作為用戶端** - 如果程式執行於某一部 iSeries 伺服器上，並存取另一部 iSeries 伺服器中的資料，您最好的選擇可能就是使用 IBM Toolbox for Java 類別。這些類別會使您輕易地存取第二部 iSeries 伺服器中的資源。

  這種存取的範例爲「資料佇列」存取。IBM Toolbox for Java 授權程式的「資料佇列」介面能使您輕易地存取資料佇列資源。

  使用 IBM Toolbox for Java 也意味著您的程式會同時在用戶端與伺服器中運作，以存取 iSeries 伺服器上的資料佇列。當執行於某一部 iSeries 伺服器上時，它也會運作，以存取另一部 iSeries 伺服器上的資料佇列。

  另一個選擇方案就是撰寫個別程式 (例如，以 C 語言撰寫)，來存取資料佇列。當 Java 程式需要存取資料佇列時，它會呼叫此程式。

  此方法較具伺服器可攜性；您可具有一個處理資料佇列存取的 Java 程式，並對每一個支援的伺服器具有不同的程式版本。

在 i5/OS Java 虛擬機器上執行 IBM Toolbox for Java 類別

在 IBM Developer Kit for Java (i5/OS) Java 虛擬機器 (JVM) 上執行 IBM Toolbox for Java 類別有一些特殊考量。

指令呼叫

用來呼叫指令的兩種常用方法如下：

- IBM Toolbox for Java CommandCall 類別
- 使用 java.lang.Runtime.exec 方法

CommandCall 類別會產生訊息清單，以便指令完成後可供 Java 程式使用。無法透過 java.lang.Runtime.exec() 來產生此訊息清單。

許多平台上都可以使用 java.lang.Runtime.exec 方法，所以如果程式須在不同類型伺服器上存取檔案，java.lang.Runtime.exec() 將會是比較理想的選擇。

整合檔案系統

以下是在 iSeries 伺服器的整合檔案系統中存取檔案的常用方法：

- IBM Toolbox for Java 授權程式的 IFSFile 類別
- 本身爲 java.io 一部分的檔案類別

IBM Toolbox for Java 整合檔案系統類別的優點在於它所提供的函數比 java.io 類別還要多。IBM Toolbox for Java 類別也可在 Applet 中運作，而且它們不需要重新導向的方法 (例如 iSeries Access for Windows) 即可從工作站到達伺服器。

java.io 類別具有可攜性而可在多個平台上執行，這就是它的優點。如果您的程式必須在不同類型的伺服器上存取檔案，java.io 是較佳的解決方案。

如果在用戶端上使用 java.io 類別，您需要重新導向的方法 (例如 iSeries Access for Windows) 方可到達伺服器檔案系統。

JDBC

有兩個 IBM 提供的 JDBC 驅動程式可供 i5/OS JVM 中執行的程式使用：

- IBM Toolbox for Java JDBC 驅動程式
- IBM Developer Kit for Java JDBC 驅動程式

當程式在主從架構環境中執行時，最好使用 IBM Toolbox for Java JDBC 驅動程式。

當程式在 iSeries 伺服器中執行時，最好使用 IBM Developer Kit for Java JDBC 驅動程式。

如果同一個程式同時在工作站與伺服器上執行，則您應透過系統內容載入正確的驅動程式，而不是在程式中編寫驅動程式的名稱。

程式呼叫

底下是呼叫程式的兩種常用方式：

- IBM Toolbox for Java 的 ProgramCall 類別
- 透過「Java 原生介面 (JNI)」呼叫

IBM Toolbox for Java 授權程式的 ProgramCall 類別，具有可呼叫任何 iSeries 伺服器程式的優點。

您可能無法透過 JNI 來呼叫 iSeries 伺服器程式。JNI 的優點就是具有可攜性而可在多個伺服器平台上執行。

在 i5/OS Java 虛擬機器中使用 AS400 物件設定系統名稱、使用者 ID 及密碼

當 Java 程式在 IBM Developer Kit for Java (i5/OS) Java 虛擬機器 (JVM) 中執行時，AS400 物件允許以特殊值代表系統名稱、使用者 ID 及密碼。

在 i5/OS JVM 中執行程式時，請留意某些特殊值及其他注意事項：

- 程式於伺服器上執行時，系統會停用使用者 ID 及密碼的提示。 伺服器環境中使用者 ID 和密碼的其它相關資訊，請參閱 AS400 物件中的使用者 ID 和密碼摘要。
- 如果未在 AS400 物件上設定系統名稱、使用者 ID 或密碼，AS400 物件會使用啓動 Java 程式之工作的使用者 ID 及密碼，與目前的伺服器連接。**在連接到 v4r3 或更早版本的機器時，必須在使用記錄層次存取時提供密碼。連線到 v4r4 或更新版本的機器時，它可以像其餘的 IBM Toolbox for Java 元件那樣，延伸登入使用者的密碼。**

- 特殊值 **localhost** 可作爲系統名稱。在此情形下，AS400 物件會連接至目前的伺服器。
- 特殊值 ***current** 可用來當作 AS400 物件中的使用者 ID 或密碼。在此情形下，將使用啓動 Java 程式之工作的使用者 ID 或密碼 (或同時使用兩者)。*current 的其它相關資訊，請參閱以下的附註。
- 當 Java 程式在某一 iSeries 伺服器的 i5/OS JVM 上執行，且程式正存取另一台 iSeries 伺服器的資源時，特殊值 ***current** 可以當作 AS400 物件中的使用者 ID 或密碼使用。在此情形下，當連接至目標系統時，將使用已在原始系統上啓動 Java 程式之工作的使用者 ID 及密碼。*current 的其它相關資訊，請參閱以下的附註。

  註：

  – 如果您使用的是記錄層次存取及 V4R3 或之前的版本，則 Java 程式不能將密碼設爲 "*current"。當您使用記錄層次存取時，"localhost" 可用於系統名稱，而 "*current" 可用於使用者 ID；不過，Java 程式必須提供密碼。

  – *current 僅能在執行版本 4 版次 3 (V4R3) 及更新的版本的系統中運作。 在執行 V4R2 的系統中，必須指定密碼及使用者 ID。

範例

下列範例將說明 AS400 物件如何與 i5/OS JVM 一起使用。

範例：在 i5/OS JVM 執行 Java 程式時建立 AS400 物件

在 i5/OS JVM 中執行 Java 程式時，此程式不需要提供系統名稱、使用者 ID 或密碼。

註: 使用記錄層次存取時，您**必須**提供密碼。

如果未提供這些值，AS400 物件將使用已啓動 Java 程式之工作的使用者 ID 及密碼，連接至本端系統。

程式執行於 i5/OS JVM 上時，將系統名稱設爲 **localhost** 與未設定系統名稱是一樣的。下列範例說明如何連接至目前的伺服器：

```
// Create two AS400 objects.  If the Java program is running in the
// i5/OS JVM, the behavior of the two objects is the same.
// They will connect to the current server using the user ID and
// password of the job that started the Java program.
AS400 sys  = new AS400()
AS400 sys2 = new AS400("localhost")
```

範例：使用與啓動工作的程式不同的使用者 ID 及密碼來連接現行伺服器 即使是在 i5/OS JVM 上執行 Java 程式，此程式仍然可以設定使用者 ID 及密碼。這些值將置換已啓動 Java 程式之工作的使用者 ID 及密碼。

在下列範例中，Java 程式將連接至現行伺服器，但程式使用的使用者 ID 及密碼不同於啓動 Java 程式之工作的使用者 ID 及密碼。

```
// Create an AS400 object. Connect to the current server but do
// not use the user ID and password of the job that started the
// program. The supplied values are used.
AS400 sys = new AS400("localhost", "USR2", "PSWRD2")
```

範例：透過啓動 Java 程式之工作的使用者 ID 及密碼，來連接另一個伺服器

在某個伺服器上執行的 Java 程式，可以連接到其他 iSeries 伺服器並使用其資源。

如果以 ***current** 代表使用者 ID 及密碼，則當 Java 程式連接至目標伺服器時，會使用啓動 Java 程式之工作的使用者 ID 及密碼。

在下列範例中，Java 程式在某一伺服器中執行，但使用另一台伺服器的資源。當程式連接至第二部伺服器時，會使用啓動 Java 程式之工作的使用者 ID 及密碼。

```
// Create an AS400 object. This program will run on one server
// but will connect to a second server (called "target").
// Because *current is used for user ID and password, the user
// ID and password of the job that started the program will be
// used when connecting to the second server.
AS400 target = new AS400("target", "*current", "*current")
```

AS/400 物件的使用者 ID 和密碼值摘要:

下表彙總在伺服器上執行的 Java 程式如何處理 AS400 物件的使用者 ID 及密碼值，以及在用戶端執行的 Java 程式又是如何處理它們的比較。

AS400 物件的值	在伺服器上執行的 Java 程式	在用戶端上執行的 Java 程式
未設定系統名稱、使用者 ID 和密碼	使用啓動程式之工作的使用者 ID 與密碼來連接現行的伺服器	提示輸人系統、使用者 ID 和密碼
系統名稱 = localhost	使用啓動程式之工作的使用者 ID 與密碼來連接現行的伺服器	**錯誤**：在用戶端上執行 Java 程式時，localhost 無效
系統名稱 = 本機主機使用者 ID = *current		
系統名稱 = 本機主機使用者 ID = *現行密碼 ID = *current		
系統名稱 = ″sys″	使用啓動程式的工作之使用者 ID 和密碼來連接伺服器「sys」使用 「sys」可以是現行伺服器或其它伺服器	提示輸入使用者 ID 和密碼
系統名稱 = 本機主機使用者 ID = ″UID″ 密碼 ID = ″PWD″	使用 Java 程式所指定的使用者 ID 及密碼連接至現行的伺服器，而非使用用來啓動程式之工作的使用者 ID 及密碼	**錯誤**：未在用戶端上執行 Java 程式時，localhost 無效

獨立輔助儲存體儲存區 (ASP)

獨立輔助儲存體儲存區 (ASP) 是一種硬碟機集合， 可獨立於系統上其餘儲存體之外，而逕行上線或離線。

獨立的 ASP 包含下列任一項：

- 一或多個使用者定義的檔案系統
- 一或多個外部檔案庫

每個獨立 ASP 都具有本身所含資料的相關必要系統資訊。因此，當系統作用中時，您可以將「獨立 ASP」離線、上線或在系統之間切換。

詳細資訊請參閱獨立 ASP 及使用者 ASP。

您可以使用 AS400JDBCDataSource 類別的 「資料庫名稱」JDBC 內容或 setDatabaseName() 方法，來指定要連接的 ASP。

所有其他 Toolbox for Java 類別 (IFSFile、Print、DataQueues 等)，都會使用連接到伺服器的使用者設定檔之工作說明所指定的「獨立 ASP」。

i5/OS 最佳化

IBM Toolbox for Java 授權程式是以 Java 撰寫，因此它可以在任何具有已認證之 Java 虛擬機器 (JVM) 的平台上執行。不管在何處，IBM Toolbox for Java 類別都依相同的方式執行。

在 iSeries JVM 上執行時，隨附於 i5/OS 的附加類別會增強 IBM Toolbox for Java 的行爲。在 iSeries JVM 上執行並連接至同一部 iSeries 伺服器時，會增進登入行爲及效能。從版本 4 版次 3 開始，i5/OS 將包含附加的類別。

啓用最佳化

IBM Toolbox for Java 分成兩種套件：單獨的授權程式及 i5/OS。

- Licensed Program 5722-JC1。授權程式版的 IBM Toolbox for Java，其檔案位在下列目錄：

  ```
  /QIBM/ProdData/http/public/jt400/lib
  ```

  這些檔案不包含最佳化的 i5/OS。如果要讓行爲與在用戶端執行 IBM Toolbox for Java 一致，請使用這些檔案。

- i5/OS。 隨附於 i5/OS 的 IBM Toolbox for Java，其檔案也在下列目錄中：

  ```
  /QIBM/ProdData/OS400/jt400/lib
  ```

  這些檔案包含在 iSeries JVM 中執行時，可最佳化 IBM Toolbox for Java 的類別。

  如需相關資訊， 請參閱 Jar 檔案相關資訊的附註 1。

登入注意事項

有了隨附於 i5/OS 的附加類別，Java 程式便具有更多的方式，可向 IBM Toolbox for Java 提供伺服器 (系統) 名稱、使用者 ID 及密碼資訊。

在存取 iSeries 伺服器上的資源時，IBM Toolbox for Java 類別必須具有系統名稱、使用者 ID 及密碼。

- **在用戶端中執行時**，是由 Java 程式提供系統名稱、使用者 ID 及密碼，或是由 IBM Toolbox for Java 透過登入對話框，從使用者擷取這些值。
- **在 iSeries Java 虛擬機器中執行時**，IBM Toolbox for Java 有另一個選擇。其可使用啓動 Java 程式之工作的使用者 ID 及密碼，傳送要求到現行 (本端) 伺服器。

有了附加的類別，在 iSeries 伺服器上執行的 Java 程式，也就能夠使用現行工作的使用者 ID 及密碼，來存取另一部 iSeries 伺服器上的資源。在此情況下，Java 程式會設定系統名稱，然後使用特殊值 ″*current″ 代替使用者 ID 及密碼。

只有當您使用的是記錄層次存取 V4R4 或更新的版本時，Java 程式才能將密碼設定爲 ″*current″。否則，當您使用記錄層次存取時，″localhost″ 可用於系統名稱，而 ″*current″ 可用於使用者 ID；不過，Java 程式必須提供密碼。

Java 程式會在 AS400 物件中設定系統名稱、使用者 ID 及密碼值。

若要使用工作的使用者 ID 及密碼，Java 程式可以使用 ″*current″ 作爲使用者 ID 及密碼，或者可以使用沒有使用者 ID 及密碼參數的建構子。

若要使用現行伺服器，Java 程式可以將 ″localhost″ 作爲系統名稱，或使用預設建構子。亦即，

```
AS400 system = new AS400();
```

同於

```
AS400 system = new AS400("localhost", "*current", "*current");
```

範例

下列範例說明如何使用最佳化的類別來登入伺服器。

範例：在使用不同 AS400 建構子的情況下登入

在下面範例中會建立兩個 AS400 物件。這兩個物件具有相同的行爲：它們都使用工作的使用者 ID 和密碼，對目前的伺服器執行指令。其中一個物件會使用特殊值代表使用者 ID 和密碼，另一個則會使用預設建構子，且不會設定使用者 ID 或密碼。

```
                    // Create an AS400 object. Since the default
                    // constructor is used and system, user ID and
                    // password are never set, the AS400 object sends
                    // requests to the local iSeries using the job's
                    // user ID and password. If this program were run
                    // on a client, the user would be prompted for
                    // system, user ID and password.
AS400 sys1 = new AS400();

                    // Create an AS400 object. This object sends
                    // requests to the local iSeries using the job's
                    // user ID and password. This object will not work
                    // on a client.
AS400 sys2 = new AS400("localhost", "*current", "*current");

                    // Create two command call objects that use the
                    // AS400 objects.
CommandCall cmd1 = new CommandCall(sys1,"myCommand1");
CommandCall cmd2 = new CommandCall(sys2,"myCommand2");

                    // Run the commands.
cmd1.run();
cmd2.run();
```

範例：使用目前工作的使用者 ID 與密碼來登入

在下列範例中，將建立一個 AS400 物件，代表第二部 iSeries 伺服器。因爲使用了 *current，所以會在第二個 (目標) 伺服器上，使用來自執行 Java 程式之 iSeries 伺服器上工作的使用者 ID 及密碼。

```
                    // Create an AS400 object. This object sends
                    // requests to a second iSeries using the user ID
                    // and password from the job on the current server.
AS400 sys = new AS400("mySystem.myCompany.com", "*current", "*current");


                    // Create a command call object to run a command
                    // on the target server.
CommandCall cmd = new CommandCall(sys,"myCommand1");


                    // Run the command.
cmd.run();
```

效能的增進

利用 i5/OS 所提供的附加類別，在 iSeries Java 虛擬機器中執行的 Java 程式可提高效能。在某些情況下會因爲使用的通訊功能減少而改善效能，但在某些情況下，會使用 iSeries API，而非呼叫伺服器程式。

較短的下載時間

爲了使 IBM Toolbox for Java 類別檔案的下載數量降到最低，請搭配使用 proxy 伺服器與 AS400ToolboxJarMaker 工具。

較快的通訊

對所有 IBM Toolbox for Java 功能 (JDBC 及整合檔案系統存取除外) 而言，在 iSeries Java 虛擬機器中執行的 Java 程式，其執行速度會更快。程式的執行速度之所以會更快，是因爲在 Java 程式與伺服器中執行要求的伺服器程式之間進行通訊時，所使用的通訊碼較少。

JDBC 與整合檔案系統存取並未最佳化，因爲已存在使這些功能的執行速度更快的機能。執行於 iSeries 之上時，您可以使用 iSeries 的 JDBC 驅動程式，代替 IBM Toolbox for Java 所隨附的 JDBC 驅動程式。若要存取伺服器中的檔案，您可以使用 Java.io，代替 IBM Toolbox for Java 所附的整合檔案系統存取類別。

直接呼叫 i5/OS API

下列 IBM Toolbox for Java 類別的效能可獲得改善，因爲這些類別會直接呼叫 i5/OS API，而非呼叫伺服器程式來執行要求：

- AS400Certificate 類別
- CommandCall
- DataQueue
- ProgramCall
- 記錄層次的資料庫存取類別
- ServiceProgramCall
- UserSpace

僅在使用者 ID 及密碼符合執行 Java 程式之工作的使用者 ID 及密碼時，才能直接呼叫 API。若要改善效能，使用者 ID 及密碼必須符合啓動 Java 程式之工作的使用者 ID 及密碼。若要取得最佳結果，請使用 localhost 代表系統名稱、*current 代表使用者 ID 及 *current 代表密碼。

埠對映變更

埠對映系統已變更過，可使得埠的存取更快速。在此變更之前，埠的要求會傳送到埠對映程式。由此處，iSeries 伺服器會判斷哪個埠可用，並傳回該埠給接受的使用者。現在，您可以告知伺服器要使用哪一個埠，或指定使用預設埠。此選項可免去伺服器替您決定埠所浪費的時間。您可使用 WRKSRVTBLE 指令來檢視或變更伺服器的埠清單。

爲改善埠對映，已新增若干方法到 AS400 類別：

- getServicePort
- setServicePort
- setServicePortsToDefault

特定語言的字串變更

特定語言的字串檔案現在已隨附於 IBM Toolbox for Java 程式中，作爲類別檔案，而非內容檔。 iSeries 伺服器在類別檔案中比在內容檔中可更快速地找到訊息。ResourceBundle.getString() 現在的執行速度更快，因爲檔案儲存在電腦會進行搜尋的第一個位置。變更爲類別檔的另外一個優點爲伺服器可以更快找到字串的轉換版本。

轉換器

容許 Java 與 iSeries 之間進行更快速有效轉換的兩種類別：

- 二進位轉換器：在 Java 位元組陣列與 Java 簡式類型之間轉換。
- 字元轉換器：在 Java String 物件與 i5/OS 字碼頁之間轉換。

此外，IBM Toolbox for Java 目前也納入了本身的轉換表，內有超過 100 個常用 CCSID。先前的 IBM Toolbox for Java 延遲了幾乎所有的 Java 文字轉換。如果 Java 沒有正確的轉換表，IBM Toolbox for Java 從伺服器中下載轉換表。

IBM Toolbox for Java 會對所能辨識的任何 CCSID 執行一切文字轉換。遇到不明的 CCSID 時，會嘗試由 Java 來處理轉換作業。IBM Toolbox for Java 絕對不會嘗試從伺服器下載轉換表。這套技術使得 IBM Toolbox for Java 應用程式執行文字轉換所花的時間大幅縮短。使用者無需採取任何行動即可享受這套文字轉換新制所帶來的優點；效能增益全部都由底層轉換表產生出來。

有關「建立 Java 程式 (CRTJVAPGM)」指令的效能要訣

如果您的 Java 程式是在 iSeries Java 虛擬機器 (JVM) 中執行，則當您從 IBM Toolbox for Java .zip 檔或 .jar 檔建立 Java 程式時，您可以**大幅改善效能**。在 iSeries 指令行中輸入 **CRTJVAPGM** 指令，建立程式。(如需詳細資訊，請參閱 **CRTJVAPGM** 指令的線上說明資訊。) 利用 **CRTJVAPGM** 指令，您可儲存在 Java 應用程式啓動時所建立的 Java 程式 (且其中包含 IBM Toolbox for Java 類別)。儲存已建立的 Java 程式可讓您節省啓動處理時間。您可節省啓動處理時間，因爲每次啓動您的 Java 應用程式時，伺服器中的 Java 程式不必再重建。

如果您使用的是 V4R2 或 V4R3 版本的 IBM Toolbox for Java，則您不能對 jt400.zip 或 jt400.jar 檔案執行 **CRTJVAPGM** 指令，因爲它太大了；不過，您可以對 jt400Access.zip 檔案執行該指令。在 V4R3 中，IBM Toolbox for Java 授權程式包含一個額外的檔案：jt400Access.zip。jt400Access.zip 僅有存取類別，而沒有視覺化類別。

在 V4R5 (或之前的版本) 系統中執行 Java 應用程式時，請使用 jt400Access.zip。在 V5R1 系統中執行 Java 應用程式時，請使用 jt400Native.jar。**CRTJVAPGM** 指令已對 jt400Native.jar 執行過。

Java 國家語言支援

Java 支援一組國家語言，但它是伺服器支援之語言的子集。

當語言之間有不符的情形時，例如，如果您在使用 Java 不支援之語言的本端工作站上執行作業，則 IBM Toolbox for Java 授權程式**可能會以英文發出一些錯誤訊息**。

IBM Toolbox for Java 的服務及支援

使用下列資源取得服務及支援。

IBM Toolbox for Java 疑難排解資訊 使用此資訊，可以協助您解決使用 IBM Toolbox for Java 時遇到的問題。

JTOpen/IBM Toolbox for Java 論壇 加入使用 IBM Toolbox for Java 的 Java 程式設計師社群。參與這個論壇，可有效地取得其他 Java 程式設計師，甚至 IBM Toolbox for Java 程式開發者本身的協助與建議。

伺服器支援 使用 IBM Server 支援網站，以瞭解工具及資源的相關資訊，協助您對 iSeries 伺服器進行有效的技術規劃及支援。

軟體支援中心 使用「IBM 軟體支援中心服務」網站，以瞭解 IBM 所提供的全方位軟體支援服務。

IBM Toolbox for Java 5722-JC1 的支援服務，是依據 iSeries 軟體產品的一般條款而提供。支援服務包括程式服務、語音支援及咨詢服務。如需相關資訊，請聯絡本地的 IBM 業務代表。

關於 IBM Toolbox for Java 程式的問題可透過程式服務及語音支援取得支援，而關於應用程式設計及除錯的事項則是以諮詢服務方式支援。

除非適合下列項目，否則 IBM Toolbox for Java 應用程式介面 (API) 呼叫是以諮詢服務方式支援：

- 它的確是 Java API 錯誤，且可經由在相當簡單的程式中重新建立，來示範該錯誤。
- 它是一個要求文件闡明的問題。
- 它是一個關於範例或文件的位置的問題。

所有程式設計輔助均是由諮詢服務加以支援，包括 IBM Toolbox for Java 授權程式中所提供的程式碼範例在內。

其他範例可在網際網路上的 iSeries 首頁  取得，這些範例是以非支援的方式提供。

「IBM Toolbox for Java 授權程式產品」附有問題解決資訊。如果您認爲 IBM Toolbox for Java API 中可能有錯誤，則需要一個可示範該錯誤的簡單程式。

程式碼範例

下列清單所提供的鏈結，可連至 IBM Toolbox for Java 資訊所使用之眾多範例的進入點。

Access 類別
Bean
Commtrace 類別
Graphical Toolbox
HTML 類別
PCML
ReportWriter 類別
資源類別
RFML
Security 類別
Servlet 類別
簡式範例
程式設計秘訣
ToolboxMe for iSeries
公用程式類別
Vaccess 類別
XPCML

IBM 授與您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的函數，來符合您的需要。

| 除法律規定不得排除的保證外，IBM、IBM 之程式開發人員及供應商不附具任何明示或默示之保證，包含且不
| 限於任何相關技術支援之未侵害他人權利之保證、或可商用性及符合特定效用等之默示保證。

| 在任何情況下，IBM、IBM 之程式開發者或供應商對下列情事均不負賠償責任，即使被告知該情事有可能發生
| 時，亦同：

| 1. 資料之滅失或毀損；
| 2. 直接、特殊、附帶或間接的傷害或其他衍生之經濟損害；或
| 3. 利潤、營業、收益、商譽或預期節餘等項之損失。

| 倘法律規定不得排除或限制賠償責任時，則該排除或限制無效。

範例：Access 類別

本節會列出 IBM Toolbox for Java Access 類別的文件所提供的所有程式碼範例。

AS400JPing

- 範例：在 Java 程式中使用 AS400JPing

BidiTransform

- 範例：使用 AS400BidiTransform 類別來轉換雙向文字

CommandCall

- 範例：使用 CommandCall 來執行伺服器上的指令
- 範例：使用 CommandCall 提示伺服器的名稱、要執行的指令及列印結果

ConnectionPool

- 範例：使用 AS400ConnectionPool 建立伺服器的連線

DataArea

- 範例：建立及寫入到小數資料區

資料轉換與說明

- 範例：使用 FieldDescription、RecordFormat 和 Record 類別
- 範例：將資料放在佇列上
- 範例：從佇列讀取資料
- 範例：搭配使用 AS400DataType 類別與 ProgramCall

DataQueue

- 範例：如何建立 DataQueue 物件、讀取資料及切斷連線
- 範例：將資料放在佇列上
- 範例：從佇列讀取資料
- 範例：使用 KeyedDataQueue 將項目放置於佇列中
- 範例：使用 KeyedDataQueue 從佇列中取出項目

數位認證

- 範例：列出屬於使用者的數位認證

EnvironmentVariable

- 範例：建立、設定及取得環境變數

異常情況

- 範例：抓取一個丟出的異常情況、擷取回覆碼及顯示異常情況文字

FTP

- 範例：使用 FTP 類別自伺服器目錄中，複製一組檔案
- 範例：使用 AS400FTP 類別自目錄中複製一組檔案

整合檔案系統

- 範例：使用 IFSFile
- 範例：使用 IFSFile.listFiles() 方法列出目錄內容
- 範例：使用 IFSFile 類別複製檔案
- 範例：使用 IFSFile 類別列出目錄內容
- 範例：如何使用 IFSJavaFile 代替 java.io.File
- 範例：使用 IFSFile 類別列出伺服器上的目錄內容

JavaApplicationCall

- 範例：自用戶端執行伺服器上的程式，輸出 ″Hello World!″

JDBC

- 範例：使用 JDBC 驅動程式來建立及擴大表格。
- 範例：使用 JDBC 驅動程式來查詢表格並輸出它的內容。

工作

- 範例：使用快取記憶體擷取及變更工作資訊
- 範例：列出所有作用中的工作
- 範例：列印工作日誌中特定使用者的所有訊息
- 範例：列出特定使用者的工作識別資訊
- 範例：取得伺服器上的工作清單，並列出工作狀態及工作識別字
- 範例：顯示工作日誌中屬於現行使用者的工作的訊息

訊息佇列

- 範例：如何使用訊息佇列物件
- 範例：列印訊息佇列的內容
- 範例：擷取及列印訊息
- 範例：列出訊息佇列的內容
- 範例：使用 Using AS400Message 與 CommandCall
- 範例：使用 Using AS400Message 與 ProgramCall

NetServer

- 範例：使用 NetServer 物件變更 NetServer 的名稱

列印

- 範例：以非同步方式列出使用 PrintObjectListListener 介面的所有排存檔
- 範例：以非同步方式列出系統中所有的排存檔，不使用 PrintObjectListListener 介面
- 範例：使用 SpooledFile.copy() 來複製排存檔
- 範例：從輸入串流建立排存檔
- 範例：使用 SCS3812Writer 類別產生一個 SCS 資料串流
- 範例：讀取現存的排存檔
- 範例：讀取及轉換排存檔
- 範例：以同步方式列出所有排存檔

許可權

- 範例：設定 AS400 物件的權限

程式呼叫

- 範例：使用 ProgramCall
- 範例：使用 ProgramCall 來擷取系統狀態
- 範例：傳送具有程式參數物件的參數資料

QSYSObjectPathName

- 範例：建置一個整合檔案系統名稱
- 範例：使用 QSYSObjectPathName.toPath() 建置一個 AS400 物件名稱
- 範例：使用 QSYSObjectPathName 來剖析整合檔案系統路徑名稱

記錄層次存取

- 範例：循序存取檔案
- 範例：使用記錄層次存取類別來讀取檔案
- 範例：使用記錄層次存取類別，依索引來讀取記錄
- 範例：使用 LineDataRecordWriter 類別

服務程式呼叫

- 範例：使用 ServiceProgramCall 呼叫程序

SystemStatus

- 範例：以 SystemStatus 類別使用快取記憶體

SystemPool

- 範例：設定 SystemPool 的最大錯誤大小

SystemValue

- 範例：使用 SystemValue 及 SystemValueList

Trace

- 範例：使用 Trace.setTraceOn() 方法
- 範例：使用追蹤的偏好方式
- 範例：使用元件追蹤

UserGroup

- 範例：擷取使用者的清單
- 範例：列出群組的所有使用者

UserSpace

- 範例：如何建立使用者空間

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：使用 CommandCall

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////
//
// Command call example.This program prompts the user
// for the name of the server and the command to run, then
// prints the result of the command.
//
// This source is an example of IBM Toolbox for Java "CommandCall"
//
////////////////////////////////////////////////////////////////////////////



import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class CommandCallExample extends Object
{
    public static void main(String[] parmeters)
   {

      // Created a reader to get input from the user
      BufferedReader inputStream = new BufferedReader(new InputStreamReader(System.in),1);



      // Declare variables to hold the system name and the command to run
      String systemString= null;
      String commandString = null;

            System.out.println( " " );



      // Get the system name and the command to run from the user
       try
      {
         System.out.print("System name: ");
          systemString = inputStream.readLine();

         System.out.print("Command: ");
          commandString = inputStream.readLine();
      }
       catch (Exception e) {};

            System.out.println( " " );



       // Create an AS400 object.
This is the system we send the command to
      AS400 as400 = new AS400(systemString);



      // Create a command call object specifying the server that will
      // recieve the command.
      CommandCall command = new CommandCall( as400 );



       try
      {
```

```
          // Run the command.
          if (command.run(commandString))
             System.out.print( "Command successful" );
         else
             System.out.print( "Command failed" );



          // If messages were produced from the command, print them
          AS400Message[] messagelist = command.getMessageList();

         if (messagelist.length > 0)
         {
             System.out.println( ", messages from the command:" );
                   System.out.println( " " );
         }

          for (int i=0; i < messagelist.length; i++)
         {
            System.out.print( messagelist[i].getID() );
            System.out.print  ( ": " );
            System.out.println( messagelist[i].getText() );
         }
      }
      catch (Exception e)
      {
         System.out.println( "Command " + command.getCommand() + " did not run" );
      }

             System.exit(0);
   }
}
```

範例：使用 AS400ConnectionPool

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////////
//
// AS400ConnectionPooling example.  This program uses an AS400ConnectionPool to
// create connections to an iSeries server.
// Command syntax:
//    AS400ConnectionPooling system myUserId myPassword
//
// For example,
//    AS400ConnectionPooling MySystem MyUserId MyPassword
//
////////////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;

public class AS400ConnectionPooling
{
   public static void main(String[] parameters)
   {
      // Check the input parameters.
            if (parameters.length != 3)
      {
                  System.out.println("");
                  System.out.println("Usage:");
                  System.out.println("");
                  System.out.println("   AS400ConnectionPooling system userId password");
                  System.out.println("");
                  System.out.println("");
         System.out.println("For example:");
                  System.out.println("");
                  System.out.println("");
```

```
            System.out.println("   AS400ConnectionPooling MySystem MyUserId MyPassword");
            System.out.println("");
        return;
}

String system   = parameters[0];
String userId   = parameters[1];
String password = parameters[2];

try
{
   // Create an AS400ConnectionPool.
   AS400ConnectionPool testPool = new AS400ConnectionPool();

   // Set a maximum of 128 connections to this pool.
   testPool.setMaxConnections(128);

   // Set a maximum lifetime for 30 minutes for connections.
   testPool.setMaxLifetime(1000*60*30);     // 30 min  Max lifetime since created.

   // Preconnect 5 connections to the AS400.COMMAND service.
   testPool.fill(system, userId, password, AS400.COMMAND, 1);
   System.out.println();
   System.out.println("Preconnected 1 connection to the AS400.COMMAND service");

   // Call getActiveConnectionCount and getAvailableConnectionCount to see how many
   // connections are in use and available for a particular system.
   System.out.println("Number of active connections: "
                      + testPool.getActiveConnectionCount(system, userId));
   System.out.println("Number of available connections for use: "
                      + testPool.getAvailableConnectionCount(system, userId));

   // Create a connection to the AS400.COMMAND service. (Use the service number
   // constants defined in the AS400 class (FILE, PRINT, COMMAND, DATAQUEUE, and so on.))
   // Since connections have already been filled, the usual time spent connecting
   // to the command service is avoided.
   AS400 newConn1 = testPool.getConnection(system, userId, password, AS400.COMMAND);

   System.out.println();
   System.out.println("getConnection gives out an existing connection to user");
   System.out.println("Number of active connections: "
                      + testPool.getActiveConnectionCount(system, userId));
   System.out.println("Number of available connections for use:  "
                      + testPool.getAvailableConnectionCount(system, userId));

   // Create a new command call object and run a command.
   CommandCall cmd1 = new CommandCall(newConn1);
   cmd1.run("CRTLIB FRED");

   // Return the connection to the pool.
   testPool.returnConnectionToPool(newConn1);

   System.out.println();
   System.out.println("Returned a connection to pool");
   System.out.println("Number of active connections: "
                      + testPool.getActiveConnectionCount(system, userId));
   System.out.println("Number of available connections for reuse: "
                      + testPool.getAvailableConnectionCount(system, userId));

   // Create a connection to the AS400.COMMAND service. This will return the same
   // object as above for reuse.
   AS400 newConn2 = testPool.getConnection(system, userId, password, AS400.COMMAND);

   System.out.println();
   System.out.println("getConnection gives out an existing connection to user");
   System.out.println("Number of active connections: "
                      + testPool.getActiveConnectionCount(system, userId));
```

```
            System.out.println("Number of available connections for reuse: "
                               + testPool.getAvailableConnectionCount(system, userId));

            // Create a connection to the AS400.COMMAND service. This will create a new
            // connection as there are not any connections in the pool to reuse.
            AS400 newConn3 = testPool.getConnection(system, userId, password, AS400.COMMAND);

            System.out.println();
            System.out.println("getConnection creates a new connection because there are no
                               connections available");
            System.out.println("Number of active connections: "
                               + testPool.getActiveConnectionCount(system, userId));
            System.out.println("Number of available connections for reuse: "
                               + testPool.getAvailableConnectionCount(system, userId));

            // Close the test pool.
            testPool.close();
         }
         catch (Exception e)
         {
            // If any of the above operations failed say the pool operations failed
            // and output the exception.

            System.out.println("Pool operations failed");
            System.out.println(e);
            e.printStackTrace();
         }
      }
   }
```

範例：使用 FieldDescription、RecordFormat 與 Record 類別

下列範例說明如何使用 FieldDescription、RecordFormat 及 Record 類別與資料佇列。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

範例：使用 FieldDescription 類別

您可使用 FieldDescription 類別來說明組成資料佇列上登錄的不同資料類型。這些範例假設資料佇列上的登錄爲下列格式：

```
Message numberSenderTime sentMessage text Reply required
      |             |           |               |               |
   bin(4)        char(50)    char(8)         char(1024)        char(1)


// Create field descriptions for the entry data
BinaryFieldDescription msgNumber = new BinaryFieldDescription(new AS400Bin4(),
                                   "msgnum");
CharacterFieldDescription sender = new CharacterFieldDescription(new AS400Text(50),
                                   "sender");
CharacterFieldDescription timeSent = new CharacterFieldDescription(new AS400Text(8),
                                     "timesent");
CharacterFieldDescription msgText = new CharacterFieldDescription(new AS400Text(1024),
                                    "msgtext");
CharacterFieldDescription replyRequired = new CharacterFieldDescription(new AS400Text(1),
                                          "replyreq");
```

使用 RecordFormat 類別

您可使用 RecordFormat 類別來說明組成資料佇列登錄的資料。

範例：定義及動態使用 RecordFormat

下列範例使用 RecordFormat 類別來說明資料佇列登錄的格式，然後動態地使用它以擷取記錄：

```
RecordFormat entryFormat = new RecordFormat();
// Describe the fields in an entry on the data queue
entryFormat.addFieldDescription(msgNumber);
entryFormat.addFieldDescription(sender);
entryFormat.addFieldDescription(timeSent);
entryFormat.addFieldDescription(msgText);
entryFormat.addFieldDescription(replyRequired);

// Get a record based on the format of the entries on the data queue
Record rec = entryFormat.getNewRecord();
```

範例：靜態定義 RecordFormat

下列範例在靜態下定義記錄格式，可允許許多程式不必多次撰寫記錄格式程式碼便可使用該格式。

```
public class MessageEntryFormat extends RecordFormat
{
  // The field descriptions are contained in the class
  static BinaryFieldDescription msgNumber = new BinaryFieldDescription(new AS400Bin4(),
                                            "msgnum");
  static CharacterFieldDescription sender = new CharacterFieldDescription(new AS400Text(50),
                                            "sender");
  static CharacterFieldDescription timeSent = new CharacterFieldDescription(new AS400Text(8),
                                              "timesent");
  static CharacterFieldDescription msgText = new CharacterFieldDescription(new AS400Text(1024),
                                             "msgtext");
  static CharacterFieldDescription replyRequired = new CharacterFieldDescription(new AS400Text(1),
                                                   "replyreq");

  public MessageEntryFormat()
  {
    // We will name this format for posterity
    super("MessageEntryFormat");
    // Add the field descriptions
    addFieldDescription(msgNumber);
    addFieldDescription(sender);
    addFieldDescription(timeSent);
    addFieldDescription(msgText);
    addFieldDescription(replyRequired);
  }
}
```

範例：靜態使用 RecordFormat

下列範例說明 Java 程式如何使用靜態定義的 Recordformat：

```
MessageEntryFormat entryFormat = new MessageEntryFormat();
// Get a record based on the format of the entries on the data queue
Record rec = entryFormat.getNewRecord();
```

使用 Record 類別

您可使用 Record 類別來存取資料佇列登錄的個別欄位。

範例：使用同屬 Record 物件

```
// Instantiate our data queue object
DataQueue dq = new DataQueue(new AS400(), "/qsys.lib/mylib.lib/myq.dtaq");

// Read an entry
DataQueueEntry dqEntry = null;
 try
{
  dqEntry = dq.read();
```

```
  }
  catch (Exception e)
  {
    // Handle any exceptions
  }

  // Get a record object from our record format, initializing it with the data from the entry we
  // just read.
  Record rec = entryFormat.getNewRecord(dqEntry.getData());

  // Output the complete entry as a String.The contents of the record are converted to Java Objects
// based on the record format of the entry.
  System.out.println(rec.toString());
  // Get the contents of the individual fields of the entry.Each field's contents are converted to
// a Java Object.
  Integer num = (Integer)rec.getField(0);// Retrieve contents by index
  String s = (String)rec.getField("sender");// Retrieve contents by field name
  String text = (String)rec.getField(3);// Retrieve the message text
  // Output the data
  System.out.println(num + " " + s + " " + text);
```

範例：使用特定的 Record 物件

您也可靜態地定義及使用特定於此資料佇列格式的 Record， 這可讓您對欄位命名所提供的 get() 及 set() 方法，比 getField() 及 setField() 更具意義。同時，藉由使用靜態定義的特定 Record，您可傳回基本的 Java 類型來代替物件，還可識別您使用者的傳回類型。

請注意，此範例必須明確地強制轉型正確的 Java 物件。

```
  public class MessageEntryRecord extends Record
  {
    static private RecordFormat format = new MessageEntryFormat();

    public MessageEntryRecord()
    {
      super(format);
    }

    public int getMessageNumber()
    {
      // Return the message number as an int.Note: We know our record format and therefore
// know the names of our fields.It is safest to get the field by name in case a field
// has been inserted into the format unbeknownst to us.
      return ((Integer)getField("msgnum")).intValue();
    }

    public String getMessageText()
    {
      // Return the text of the message
      return (String)getField("msgtext");
    }

    public String getSender()
    {
      // Return the sender of the message
      return (String)getField("sender");
    }

    public String getTimeSent()
    {
      // Return the sender of the message
      return (String)getField("timesent");
    }

    // We could add setters here
  }
```

範例：傳回新的 MessageEntryRecord

我們需要置換 MessageEntryFormat 類別 (於前一範例中)若要置換方法，請在 MessageEntryFormat 類別中新增如下：

```
    public Record getNewRecord(byte[] data)
    {
      Record r = new MessageEntryRecord();
      r.setContents(data);
      return r;
    }
```

新增新的 getNewRecord() 方法後，您可使用 MessageEntryRecord 來解譯資料佇列登錄：

```
  // Get a record object from our record format, initializing it with the data from the entry we
  // just read.
Note the use of the new overridden method getNewRecord().
  MessageEntryRecord rec = (MessageEntryRecord)entryFormat.getNewRecord(dqEntry.getData());

  // Output the complete entry as a String.The contents of the record are converted to Java Objects
// based on the record format of the entry.
  System.out.println(rec.toString());
  // Get the contents of the individual fields of the entry.Each field's contents are converted to
// a Java Object.
  int num = rec.getMessageNumber();// Retrieve the message number as an int
  String s = rec.getSender(); // Retrieve the sender
  String text = rec.getMessageText(); // Retrieve the message text
  // Output the data
  System.out.println(num + " " + s + " " + text);
```

範例：使用 DataQueue 類別以在佇列上放置資料

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////
//
// Data Queue example.This program uses the DataQueue class to put
// records on a data queue.
//
// This example uses the Record and Record format classes to put data
// on the queue.String data is converted from Unicode to ebcdic
// and numbers are converted from Java to the server format.Since data
// is converted the data queue entries can be read by a server program
// or a iSeries Access for Windows program as well as another Java program.
//
// This is the producer side of the producer/consumer example.  It puts work
// items on the queue for the consumer to process.
//
// Command syntax:
//DQProducerExample system
//
////////////////////////////////////////////////////////////////////////////


import java.io.*;
 import java.util.*;
import java.net.*;
import com.ibm.as400.access.*;

public class DQProducerExample extends Object
{
    // Create a reader to get input from the user.
    static BufferedReader inputStream = new BufferedReader(new InputStreamReader(System.in),1);
public static void main(String[] parameters)
   {
            System.out.println( " " );
```

```
// if the system name was not specified, display help text and exit.
if (parameters.length >= 1)
{
    try
   {
       // The first parameter is the system that contains the data queue.
       String system = parameters[0];

       // Create an AS400 object for the server that has the data queue.
       AS400 as400 = new AS400(system);

       // Build a record format for the format of the data queue entry.
       // This format matches the format in the DQConsumer class.  A
       // record consists of:
       //    - a four byte number    -- the customer number
       //    - a four byte number    -- the part number
       //    - a 20 character string -- the part description
       //    - a four byte number    -- the number of parts in this order
       // First create the base data types.
       BinaryFieldDescription customerNumber =
          new BinaryFieldDescription(new AS400Bin4(), "CUSTOMER_NUMBER");

       BinaryFieldDescription partNumber =
          new BinaryFieldDescription(new AS400Bin4(), "PART_NUMBER");

       CharacterFieldDescription partName =
          new CharacterFieldDescription(new AS400Text(20, as400), "PART_NAME");

       BinaryFieldDescription quantity =
          new BinaryFieldDescription(new AS400Bin4(), "QUANTITY");

       // Build a record format and fill it with the base data types.
       RecordFormat dataFormat = new RecordFormat();
       dataFormat.addFieldDescription(customerNumber);
       dataFormat.addFieldDescription(partNumber);
       dataFormat.addFieldDescription(partName);
       dataFormat.addFieldDescription(quantity);

       // Create the library that contains the data queue
       // using CommandCall.
       CommandCall crtlib = new CommandCall(as400);
       crtlib.run("CRTLIB JAVADEMO");

       // Create the data queue object.
       DataQueue dq = new DataQueue(as400, "/QSYS.LIB/JAVADEMO.LIB/PRODCONS.DTAQ");

       // Create the data queue just in case this is the first time this
       // program has run.  The queue already exists exception is caught
       // and ignored.
       try
       {
          dq.create(96);
       }
       catch (Exception e) {};

       // Get the first field of data from the user.
       System.out.print("Enter customer number (or 0 to quit): ");
       int customer = getInt();

       // While there is data to put on the queue.
       while (customer > 0)
       {
          // Get the rest of the data for this order from the user.
          System.out.print("Enter part number: ");
          int part = getInt();
```

```
                System.out.print("Enter quantity: ");
                int quantityToOrder = getInt();

                String description = "part " + part;

                // Create a record based on the record format.The record
// is empty now but will eventually contain the data.
                Record data = new Record(dataFormat);

                // Set the values we received from the user into the record.
                data.setField("CUSTOMER_NUMBER", new Integer(customer));
                data.setField("PART_NUMBER",     new Integer(part));
                data.setField("QUANTITY",        new Integer(quantityToOrder));
                data.setField("PART_NAME", description);

                // Convert the record into a byte array.  The byte array is
                // what is actually put to the data queue.
                byte [] byteData = data.getContents();

                System.out.println("");
                System.out.println("Writing record to the server ...");
                System.out.println("");

                // Write the record to the data queue.
                dq.write(byteData);

                // Get the next value from the user.
                System.out.print("Enter customer number (or 0 to quit): ");
                customer = getInt();
             }
          }
          catch (Exception e)
          {
             // If any of the above operations failed say the data queue
             // operation failed and output the exception.

             System.out.println("Data Queue operation failed");
             System.out.println(e);
          }
       }

       // Display help text when parameters are incorrect.
       else
       {
          System.out.println("");
          System.out.println("");
          System.out.println("");
          System.out.println("Parameters are not correct. Command syntax is:");
          System.out.println("");
          System.out.println(" DQProducter system");
          System.out.println("");
          System.out.println("Where");
          System.out.println("");
          System.out.println(" system = Server that has the data queue");
          System.out.println("");
          System.out.println("For example:");
          System.out.println("");
          System.out.println(" DQProducerExample mySystem");
          System.out.println("");
          System.out.println("");
       }

       System.exit(0);
    }

    // This is the subroutine that gets a character string from the user
    // and converts it into an int.
```

```
   static int getInt()
   {
      int i = 0;
      boolean Continue = true;

      while (Continue)
      {
         try
         {
            String s = inputStream.readLine();

            i = (new Integer(s)).intValue();
            Continue = false;
         }
         catch (Exception e)
         {
            System.out.println(e);
            System.out.print("Please enter a number ==>");
         }
      }

      return i;
   }
}
```

範例：使用 DataQueue 類別以讀取資料佇列登錄

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////
//
// Data Queue example.This program uses the Data Queue classes to read
// entries off a data queue on the server.  The entries were put on the
// queue with the DQProducer example program.
//
// This is the consumer side of the producer/consumer example.It reads
// entries off the queue and process them.
//
// Command syntax:
//DQConsumerExample system
//
////////////////////////////////////////////////////////////////////////////


import java.io.*;
 import java.util.*;
import java.net.*;
import com.ibm.as400.access.*;

public class DQConsumerExample extends Object
{
   public static void main(String[] parameters)
   {
            System.out.println( " " );

      // if a system name was not specified, display help text and exit.
      if (parameters.length >= 1)
      {
          try
         {

             // The first parameter is the system that contains the data queue.
             String system = parameters[0];

             // Create an AS400 object for the server that has the data queue.
             AS400 as400 = new AS400(system);
```

```
            // Build a record format for the format of the data queue entry.
            // This format matches the format in the DQProducer class.  A
            // record consists of:
            //    - a four byte number -- the customer number
            //    - a four byte number -- the part number
            //    - a 20 character string -- the part description
            //    - a four byte number -- the number of parts in this order

            // First create the base data types.
            BinaryFieldDescription customerNumber =
               new BinaryFieldDescription(new AS400Bin4(), "CUSTOMER_NUMBER");

            BinaryFieldDescription partNumber =
               new BinaryFieldDescription(new AS400Bin4(), "PART_NUMBER");

            CharacterFieldDescription partName =
               new CharacterFieldDescription(new AS400Text(20, as400), "PART_NAME"

            BinaryFieldDescription quantity =
               new BinaryFieldDescription(new AS400Bin4(), "QUANTITY"

            // Build a record format and fill it with the base data types.
            RecordFormat dataFormat = new RecordFormat();

            dataFormat.addFieldDescription(customerNumber);
            dataFormat.addFieldDescription(partNumber);
            dataFormat.addFieldDescription(partName);
            dataFormat.addFieldDescription(quantity);

            // Create the data queue object that represents the data queue on
            // the server.
            DataQueue dq = new DataQueue(as400, "/QSYS.LIB/JAVADEMO.LIB/PRODCONS.DTAQ");

            boolean Continue = true;

            // Read the first entry off the queue.  The timeout value is
            // set to -1 so this program will wait forever for an entry.
            System.out.println("*** Waiting for an entry for process ***");

            DataQueueEntry DQData = dq.read(-1);

            while (Continue)
            {

               // We just read an entry off the queue.Put the data into
// a record object so the program can access the fields of
               // the data by name.The Record object will also convert
// the data from server format to Java format.
               Record data = dataFormat.getNewRecord(DQData.getData());

               // Get two values out of the record and display them.
               Integer amountOrdered = (Integer) data.getField("QUANTITY");
               String  partOrdered   = (String)  data.getField("PART_NAME");

               System.out.println("Need " + amountOrdered + " of "
                                  + partOrdered);
               System.out.println(" ");
               System.out.println("*** Waiting for an entry for process ***");

               // Wait for the next entry.
               DQData = dq.read(-1);
            }
         }
         catch (Exception e)
         {
            // If any of the above operations failed say the data queue
```

```
            // operation failed and output the exception.
            System.out.println("Data Queue operation failed");
            System.out.println(e);
         }
      }

      // Display help text when parameters are incorrect.
      else
      {
         System.out.println("");
         System.out.println("");
         System.out.println("");
         System.out.println("Parameters are not correct. Command syntax is:");
         System.out.println("");
         System.out.println(" DQConsumerExample system");
         System.out.println("");
         System.out.println("Where");
         System.out.println("");
         System.out.println(" system = Server that has the data queue");
         System.out.println("");
         System.out.println("For example:");
         System.out.println("");
         System.out.println(" DQConsumerExample mySystem");
         System.out.println("");
         System.out.println("");
      }

      System.exit(0);
   }
}
```

資料類型範例的用法

您可以搭配使用 AS400DataType 類別與 ProgramCall，以提供資料給程式參數，及解譯傳回到程式參數的資料。

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

範例：使用 AS400DataType 類別與 ProgramCall

下列範例將說明如何使用 ProgramCall 呼叫系統 API、 QUSRMBRD「擷取成員說明」，藉以使用 AS400DataType 類別。QUSRMBRD API 會擷取資料庫檔案中特定成員的說明。範例後的表格將列出必要的 QUSRMBRD 參數以及範例擷取的資訊類型。

```
// Create a ProgramCall object.  We will set the program name and
// parameter list later.
ProgramCall qusrmbrd = new ProgramCall(new AS400());

// Create an empty program parameter list
ProgramParameter[] parms = new ProgramParameter[6];

// Create AS400DataTypes to convert our input parameters from Java types
// to server data
AS400Bin4 bin4 = new AS400Bin4();

// We need a separate AS400Text object for each parameter with a
// different length because the AS400Text class requires the length to
// be specified.
AS400Text char8Converter = new AS400Text(8)
AS400Text char20Converter = new AS400Text(20);
AS400Text char10Converter = new AS400Text(10);
AS400Text char1Converter = new AS400Text(1);

// Populate our parameter list; we use the AS400DataType objects to
// convert our Java values to byte arrays containing server data.
```

```
    // For the output parameter we need only specify how many bytes will
    // be returned
    parms[0] = new ProgramParameter(135);
    parms[1] = new ProgramParameter(bin4.toBytes(new Integer(135)));
    parms[2] = new ProgramParameter(char8Converter.toBytes("MBRD0100"));
    parms[3] = new ProgramParameter(char20Converter.toBytes("MYFILE    MYLIB     "));
    parms[4] = new ProgramParameter(char10COnverter.toBytes("MYMEMBER  "));
    parms[5] = new ProgramParameter(char1Converter.toBytes("0"));

    // Set the program name and parameter list
    qusrmbrd.setProgram("/qsys.lib/qusrmbrd.pgm", parms);

    // Call the program
     try
    {
      qusrmbrd.run();
    }
    catch (Exception e)
    {
      // Handle any exceptions
    }

    // Get the information retrieved. Note that this is raw server data.
byte[] receiverVar = parms[0].getOutputData();


  // We need this to convert the time and date data
     AS400Text char13Converter = new AS400Text(13);

  // We need this to convert the text description data
     AS400Text char50Converter = new AS400Text(50);

     // Create an AS400Structure to handle the returned information
     AS400DataType[] dataTypeArray = new AS400DataType[11];
     dataTypeArray[0] = bin4;
     dataTypeArray[1] = bin4;
     dataTypeArray[2] = char10Converter;
     dataTypeArray[3] = char10Converter;
     dataTypeArray[4] = char10Converter;
     dataTypeArray[5] = char10Converter;
     dataTypeArray[6] = char10Converter;
     dataTypeArray[7] = char13Converter;
     dataTypeArray[8] = char13Converter;
     dataTypeArray[9] = char50Converter;
     dataTypeArray[10] = char1Converter;
     AS400Structure returnedDataConverter = new AS400Structure(dataTypeArray);


     // Convert the data returned to an array of Java Objects using our
     // returnedDataConverter
     Object[] qusrmbrdInfo = dataConverter.toObject(receiverVar, 0);


     // Get the number of bytes returned
     Integer bytesReturned = (Integer)qusrmbrdInfo[0];
     Integer bytesAvailable = (Integer)qusrmbrdInfo[1];
     if (bytesReturned.intValue() != 135)
     {
       System.out.println("Wrong amount of information returned.");
              System.exit(0);
     }
     String fileName = (String)qusrmbrdInfo[2];
     String libName = (String)qusrmbrdInfo[3];
     String mbrName = (String)qusrmbrdInfo[4];
     String fileAttribute = (String)qusrmbrdInfo[5];
     String sourceType = (String)qusrmbrdInfo[6];
     String created = (String)qusrmbrdInfo[7];
```

```
String lastChanged = (String)qusrmbrdInfo[8];
String textDesc = (String)qusrmbrdInfo[9];
String isSourceFile = (String)qusrmbrdInfo[10];

// We will just output all the information to the screen
System.out.println(fileName + " " + libName + " " + mbrName + " " +
                   fileAttribute + sourceType + " " + created + " " +
                   lastChanged + " " + textDesc + " " + isSourceFile);
```

下列表格將列出用於前述範例的 QUSRMBRD API 必要參數。

QUSRMBRD 參數	輸入或輸出	類型	說明
接收器變數	輸出	Char(*)	將包含所擷取資訊的字元緩衝區。
接收器變數的長度	輸入	Bin(4)	提供給接收器變數的字元緩衝區長度。
格式名稱	輸入	Char(8)	指定要擷取之資訊類型的格式。必須是下列其中一項： • MBRD0100 • MBRD0200 • MBRD0300 下列範例指定 MBRD0100。
限定資料庫檔案名稱	輸入	Char(20)	指限定的檔案名稱。此爲填好 10 個字元的檔案名稱空格， 後面跟著填好 10 個字元的檔案庫名稱空格。檔案庫名稱可以使用 *CURLIB 及 *LIBL 的特殊值。
資料庫成員名稱	輸入	Char(10)	填好 10 個字元的成員名稱空格。容許使用 *FIRST 及 *LAST 的特殊值。
置換處理程序	輸入	Char(1)	是否要處理置換。0 指出不處理置換。這是我們將指定的值。

下列表格將列出範例擷取的資訊類型 (根據前述範例中指定的格式 MBRD0100)：

擷取的資訊	類型
傳回的位元組	Bin(4)
可用的位元組	Bin(4)
資料庫檔案名稱	Char(10)
資料庫檔案程式庫名稱	Char(10)
成員名稱	Char(10)
檔案屬性〔檔案的類型：PF、LF 及 DDMF〕	Char(10)
來源類型〔如果此爲來源檔案，則指來源成員的類型〕	Char(10)
建立日期與時間	Char(13)
上次變更來源的日期與時間	Char(13)
成員本文說明	Char(50)

擷取的資訊	類型
來源檔案〔檔案是否爲來源檔：0=資料檔、1=來源檔〕	Char(1)

範例：使用 KeyedDataQueue

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////
//
// Data Queue example.  This program uses the KeyedDataQueue class to put
// records on a data queue.
//
// The key is a number and the data is a Unicode string.  This program
// shows one way to convert on int into a byte array and how to convert
// a Java string into a byte array so it can be written to the queue.
//
// This is the producer side of the producer/consumer example.  It puts work
// items on the queue for the consumer to process.
//
// Command syntax:
//    DQKeyedProducer system
//
////////////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;
import java.net.*;
import com.ibm.as400.access.*;

public class DQKeyedProducer extends Object
{

    // Create a reader to get input from the user.

    static BufferedReader inputStream = new BufferedReader(new InputStreamReader(System.in),1);
public static void main(String[] parameters)
   {

            System.out.println( " " );



      // if the system name was not specified, display help text and exit.

      if (parameters.length >= 1)
      {

                  // The first parameter is the system that contains the data queue.

         String system = parameters[0];



         System.out.println("Priority is a numeric value. The value ranges are:");
         System.out.println("  0 -  49 = low priority");
         System.out.println(" 50 - 100 = medium priority");
         System.out.println("100 +     = high priority");
         System.out.println(" ");

          try
         {

             // Create an AS400 object for the server that has the data queue.
```

```
AS400 as400 = new AS400(system);



// Use CommandCall to create the library that contains the
// data queue.

CommandCall crtlib = new CommandCall(as400);
crtlib.run("CRTLIB JAVADEMO");



// Create the data queue object.

QSYSObjectPathName name = new QSYSObjectPathName("JAVADEMO", "PRODCON2", "DTAQ");

KeyedDataQueue dq = new KeyedDataQueue(as400, name.getPath());



// Create the data queue just in case this is the first time this
// program has run.  The queue already exists exception is caught
// and ignored.  The length of the key is four bytes, the length
// of an entry is 96 bytes.

try
{
   dq.create(4, 96);
}
catch (Exception e) {};



// Get the data from the user.

System.out.print("Enter message: ");
String message = inputStream.readLine();

System.out.print("Enter priority: ");
int priority = getInt();



// While there is data to put on the queue.

while (priority > 0)
{

   // We want to write a java string as the entry to the queue.
   // Input the data queue is a byte array, however, so convert
   // the string to a byte array.

   byte [] byteData = message.getBytes("UnicodeBigUnmarked");



   // The key is a number.  Input to the data queue is a byte
   // array, however, so convert the int to a byte array;

   byte [] byteKey = new byte[4];
   byteKey[0] = (byte) (priority >>> 24);
   byteKey[1] = (byte) (priority >>> 16);
   byteKey[2] = (byte) (priority >>> 8);
   byteKey[3] = (byte) (priority);


   System.out.println("");
```

```
            System.out.println("Writing record to the server...");
            System.out.println("");



            // Write the record to the data queue.

            dq.write(byteKey, byteData);



            // Get the next value from the user.

            System.out.print("Enter message: ");
            message = inputStream.readLine();

            System.out.print("Enter priority: ");
            priority = getInt();
         }
      }
      catch (Exception e)
      {

         // If any of the above operations failed say the data queue
         // operation and output the exception.

         System.out.println("Data Queue operation failed");
         System.out.println(e);
      }
   }


   // Display help text when parameters are incorrect.

   else
   {
      System.out.println("");
      System.out.println("");
      System.out.println("");
      System.out.println("Parameters are not correct.  Command syntax is:");
      System.out.println("");
      System.out.println("  DQKeyedProducter system");
      System.out.println("");
      System.out.println("Where");
      System.out.println("");
      System.out.println("  system = server that has the data queue");
      System.out.println("");
      System.out.println("For example:");
      System.out.println("");
      System.out.println("  DQKeyedProducer mySystem");
      System.out.println("");
      System.out.println("");
   }

   System.exit(0);
}



// This is the subroutine that gets a character string from the user
// and converts it into an int.

static int getInt()
{
   int i = 0;
   boolean Continue = true;
```

```
      while (Continue)
      {

         try
         {
            String s = inputStream.readLine();

            i = (new Integer(s)).intValue();
            Continue = false;
         }
         catch (Exception e)
         {
            System.out.println(e);
            System.out.print("Please enter a number ==>");
         }
      }

      return i;
   }
}
```

範例：使用 KeyedDataQueue 類別以讀取資料佇列登錄

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////
//
// Keyed Data Queue example.  This program uses the KeyedDataQueue classes to
// read entries off a data queue on the server.  The entries were put on the
// queue with the DQKeyedProducer example program.
//
// The key is a number and the data is a unicode string.This program
// shows one way to convert the byte array to a Java int and to read
// a byte array and convert it into a Java string.
//
// This is the consumer side of the producer/consumer example.It reads
// entries off the queue and process them.
//
// Command syntax:
//DQKeyedConsumer system
//
////////////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;
import java.net.*;
import com.ibm.as400.access.*;

public class DQKeyedConsumer extends Object
{
   public static void main(String[] parameters)
   {
            System.out.println( " " );

      // if a system name was not specified, display help text and exit.
      if (parameters.length >= 1)
      {

                // The first parameter is the system that contains the data queue.
         String system = parameters[0];

          // Create byte arrays for the priority boundries:
          // 100 += high priority
          // 50 - 100 = medium priority
          //0 -49 = low priority
```

```
 byte [] key0 = new byte[4];
 key0[0] = 0;
 key0[1] = 0;
 key0[2] = 0;
 key0[3] = 0;

 byte [] key50= new byte[4];
key50[0] = (byte) (50 >>> 24);
key50[1] = (byte) (50 >>> 16);
key50[2] = (byte) (50 >>> 8);
 key50[3] = (byte) (50);

 byte [] key100 = new byte[4];
key100[0] = (byte) (100 >>> 24);
key100[1] = (byte) (100 >>> 16);
key100[2] = (byte) (100 >>> 8);
 key100[3] = (byte) (100);

 try
{
    // Create an AS400 object for the server that has the data queue.
    AS400 as400 = new AS400(system);

    // Create the data queue object that represents the data queue
    // on the server.

    QSYSObjectPathName name = new QSYSObjectPathName("JAVADEMO",
                                                     "PRODCON2",
                                                     "DTAQ");
    KeyedDataQueue dq = new KeyedDataQueue(as400, name.getPath());
    KeyedDataQueueEntry DQData = null;

    try
    {
       boolean Continue = true;

       // Go until the user ends us.
       while (Continue)
       {
          // Look for a high priority item on the queue. If one is
          // found process that item.  Note the peek method does not
          // remove the item if one is found. Also note the timeout
          // is 0. If an item is not found we get control back with
          // a null data queue entry.
          DQData = dq.read(key100, 0, "GE");

          if (DQData != null)
          {
             processEntry(DQData);
          }

          // else no high priority item was found.  Look for a medium
          // priority item.
          else
          {
             DQData = dq.read(key50, 0, "GE");

             if (DQData != null)
             {
                processEntry(DQData);
             }

             // else no medium priority item was found, look for a low
             // priority item.
             else
             {
                DQData = dq.read(key0, 0, "GE");
```

```
                        if (DQData != null)
                        {
                           processEntry(DQData);
                        }

                        else
                        {
                          System.out.println("Nothing to process, will check again in 30 seconds");
                          Thread.sleep(30000);
                        }
                     }
                  }
               }
            }
            catch (Exception e)
            {
               // If any of the above operations failed say the data queue
               // operation failed and output the exception.
               System.out.println("Could not read from the data queue.");
               System.out.println(e);
            };

         }
         catch (Exception e)
         {
             // If any of the above operations failed say the data queue
             // operation failed and output the exception.

             System.out.println("Data Queue operation failed");
             System.out.println(e);
         }
      }

      // Display help text when parameters are incorrect.
      else
      {
         System.out.println("");
         System.out.println("");
         System.out.println("");
         System.out.println("Parameters are not correct. Command syntax is:");
         System.out.println("");
         System.out.println(" DQKeyedConsumer system");
         System.out.println("");
         System.out.println("Where");
         System.out.println("");
         System.out.println(" system = Server that has the data queue");
         System.out.println("");
         System.out.println("For example:");
         System.out.println("");
         System.out.println("");
         System.out.println(" DQKeyedConsumer mySystem");
         System.out.println("");
         System.out.println("");
      }

      System.exit(0);
   }

   static void processEntry(KeyedDataQueueEntry DQData)
   {
      try
      {
          // The data is a string.  Get the string out of the data queue entry.
          // In the data queue entry the data is a byte array so convert the
          // entry from a byte array into a string.
          String message  = new String(DQData.getData(), "UnicodeBig");
```

```
          // The key is a byte array.Get the key out of the data queue entry
// and convert it into a number.
          byte [] keyData = DQData.getKey();

          int keyValue = ((keyData[0] & 0xFF) << 24) +
                          ((keyData[1] & 0xFF) << 16) +
                         ((keyData[2] & 0xFF) <<  8) +
                          (keyData[3] & 0xFF);

          // Output the entry.
          System.out.println("Priority: " + keyValue + "   message: " + message);

      }
      catch (Exception e)
      {
         // If any of the above operations failed say the data queue operation
         // failed and output the exception.

         System.out.println("Could not read from the data queue");
         System.out.println(e);
      }
   }
}
```

範例：使用 IFSFile

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

下列範例說明如何使用 IFSFile 類別：

- 範例：建立目錄
- 範例：使用 IFSFile 異常來追蹤錯誤
- 範例：列出副檔名是 .txt 的檔案
- 第 443 頁的『範例：使用 IFSFile listFiles() 方法以列出目錄內容』

範例：建立目錄

```
                       // Create an AS400 object.
This new
                       // directory will be created on this
                       // iSeries.
     AS400 sys = new AS400("mySystem.myCompany.com");

                       // Create a file object that
                       // represents the directory.
     IFSFile aDirectory = new IFSFile(sys, "/mydir1/mydir2/newdir");

                       // Create the directory.
     if (aDirectory.mkdir())
        System.out.println("Create directory was successful");

                       // Else the create directory failed.
     else
     {
                       // If the object already exists,
                       // find out if it is a directory or
                       // file, then display a message.
        if (aDirectory.exists())
        {
           if (aDirectory.isDirectory())
              System.out.println("Directory already exists");
           else
              System.out.println("File with this name already exists");
```

```
   }
   else
      System.out.println("Create directory failed");
}

                  // Disconnect since I am done
                  // accessing files.
sys.disconnectService(AS400.FILE);
```

範例：使用 IFSFile 異常以追蹤錯誤

發生錯誤時，IFSFile 類別會擲出 ExtendedIOException 異常。此異常情況包含一個指出失敗原因的回覆碼。即使 IFSFile 複製的 java.io 類別不丟出異常情況，IFSFile 類別也 會擲出此異常情況。例如，來自 java.io.File 的刪除方法傳回一布耳值來指出順利完成或失敗。IFSFile 中的刪除方法傳回一布耳值，但如果刪除失敗的話則會擲出 ExtendedIOException。ExtendedIOException 會將刪除失敗原因的詳細資訊提供給 Java 程式。

```
                  // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                  // Create a file object that
                  // represents the file.
IFSFile aFile = new IFSFile(sys, "/mydir1/mydir2/myfile");

                  // Delete the file.
try
{
   aFile.delete();

                  // The delete was successful.
   System.out.println("Delete successful ");
}

                  // The delete failed. Get the return
                  // code out of the exception and
                  // display why the delete failed.
catch (ExtendedIOException e)
{
   int rc = e.getReturnCode();

   switch (rc)
   {
      case ExtendedIOException.FILE_IN_USE:
         System.out.println("Delete failed, file is in use ");
         break;

      case ExtendedIOException.PATH_NOT_FOUND:
         System.out.println("Delete failed, path not found ");
         break;

                     // ... for every specific error
                     // you want to track.

      default:
         System.out.println("Delete failed, rc = ");
         System.out.println(rc);
   }
}
```

範例：列出具有 .txt 副檔名的檔案

Java 程式可以選擇性地指定在列出目錄中的檔案時的符合準則。符合的基準可以減少伺服器傳回至 IFSFile 物件的檔案數，以增進效能。下面範例說明如何列示具有副檔名 .txt 的檔案：

```
                    // Create the AS400 object.
     AS400 system = new AS400("mySystem.myCompany.com");

                    // Create the file object.
     IFSFile directory = new IFSFile(system, "/");

                    // Generate a list of all files with
                    // extension .txt
     String[] names = directory.list("*.txt");

                    // Display the names.
     if (names != null)
       for (int i = 0; i < names.length; i++)
         System.out.println(names[i]);
     else
       System.out.println("No .txt files");
```

範例：使用 IFSFile listFiles() 方法以列出目錄內容

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////////
//
// IFSListFiles example.  This program uses the integrated file system
// classes to list the contents of a directory on the server.
//
// Command syntax:
//    IFSListFiles system directory
//
// For example,
//    IFSListFiles MySystem /path1
//
////////////////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class IFSListFiles extends Object
{
    public static void main(String[] parameters)
    {
              System.out.println( " " );

              String directoryName = "";
              String system        = "";


                // if both parameters were not specified, display help text and exit.

        if (parameters.length >= 2)
        {

            // Assume the first parameter is the system name and
            // the second parameter is the directory name

            system = parameters[0];
            directoryName = parameters[1];

            try
            {
                // Create an AS400 object for the server that holds the files.

                AS400 as400 = new AS400(system);
```

```
        // Create the IFSFile object for the directory.

        IFSFile directory = new IFSFile(as400, directoryName);


        // Generate a list of IFSFiles.  Pass the listFiles method
        // the directory filter object and the search match
        // criteria.  This method caches attribute information.  For
        // instance, when isDirectory() is called on an IFSFile
        // object in the file array returned in the following code,
        // no call to the server is required.
        //
        // However, with the user of the listFiles method, attribute
        // information will not be refreshed automatically from the
        // server. This means attribute information can become
        // inconsistent with information about the server.

        IFSFile[] directoryFiles = directory.listFiles(new MyDirectoryFilter(),"*");

        // Tell the user if the directory doesn't exist or is empty

        if (directoryFiles == null)
        {
            System.out.println("The directory does not exist");
            return;
        }

        else if (directoryFiles.length == 0)
        {
            System.out.println("The directory is empty");
            return;
        }

        for (int i=0; i< directoryFiles.length; i++)
            {
                // Print out information on list.
                // Print the name of the current file

                System.out.print(directoryFiles[i].getName());


                // Pad the output so the columns line up

                for (int j = directoryFiles[i].getName().length(); j <18; j++)
                    System.out.print(" ");


                // Print the date the file was last changed.

                long changeDate = directoryFiles[i].lastModified();
                Date d = new Date(changeDate);
                System.out.print(d);
                System.out.print("  ");


                // Print if the entry is a file or directory

                System.out.print("   ");

                if (directoryFiles[i].isDirectory())
                    System.out.println("");
                else
                    System.out.println(directoryFiles[i].length());

            }
    }
```

```
            catch (Exception e)
            {
                // If any of the above operations failed say the list failed
                // and output the exception.

                System.out.println("List failed");
                System.out.println(e);
            }
        }

        // Display help text when parameters are incorrect.

        else
        {
            System.out.println("");
            System.out.println("");
            System.out.println("");
            System.out.println("Parameters are not correct.  Command syntax is:");
            System.out.println("");
            System.out.println("   IFSListFiles as400 directory");
            System.out.println("");
            System.out.println("Where");
            System.out.println("");
            System.out.println("   as400 = system that contains the files");
            System.out.println("   directory = directory to be listed");
            System.out.println("");
            System.out.println("For example:");
                        System.out.println("");
            System.out.println("   IFSListFiles mySystem /dir1/dir2");
            System.out.println("");
            System.out.println("");
        }

        System.exit(0);
    }
}


////////////////////////////////////////////////////////////////////////////
//
// The directory filter class prints information from the file object.
//
// Another way to use the filter is to simply return true or false
// based on information in the file object.  This lets the mainline
// function decide what to do with the list of files that meet the
// search criteria.
//
////////////////////////////////////////////////////////////////////////////



class MyDirectoryFilter implements IFSFileFilter
{
    public boolean accept(IFSFile file)
    {
        try
        {
            // Keep this entry.  Returning true tells the IFSList object
            // to return this file in the list of entries returned to the
            // .list() method.

            return true;
        }

        catch (Exception e)
        {
```

```
            return false;
        }
    }
}
```

範例：使用 IFS 類別以將檔案從某一目錄複製到另一目錄

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////
//
// IFSCopyFile example.This program uses the installable file system classes
// to copy a file from one directory to another on the server.
//
// Command syntax:
//IFSCopyFile system sourceName TargetName
//
// For example,
//IFSCopyFile MySystem/path1/path2/file.ext/path3/path4/path5/file.ext
//
////////////////////////////////////////////////////////////////////////////



import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class IFSCopyFile extends Object
{
   public static void main(String[] parameters)
   {
            System.out.println( " " );

      String sourceName = "";
      String targetName = "";
            String system        = "";
      byte[] buffer = new byte[1024 * 64];

      IFSFileInputStreamsource = null;
      IFSFileOutputStream target = null;

      // if all three parameters were not specified, display help text and exit.

      if (parameters.length > 2)
      {

          // Assume the first parameter is the system name,
          // the second parameter is the source name and
          // the third parameter is the target name.

         system = parameters[0];
          sourceName = parameters[1];
          targetName = parameters[2];

          try
         {
            // Create an AS400 object for the server that holds the files.

            AS400 as400 = new AS400(system);



            // Open the source file for exclusive access.

            source = new IFSFileInputStream(as400, sourceName, IFSFileInputStream.SHARE_NONE);
```

```
      System.out.println("Source file successfully opened");


      // Open the target file for exclusive access.

      target = new IFSFileOutputStream(as400, targetName, IFSFileOutputStream.SHARE_NONE, false);

      System.out.println("Target file successfully opened");


      // Read the first 64K bytes from the source file.

      int bytesRead = source.read(buffer);


      // While there is data in the source file copy the data from
      // the source file to the target file.

      while (bytesRead > 0)
      {
         target.write(buffer, 0, bytesRead);
         bytesRead = source.read(buffer);
      }

      System.out.println("Data successfully copied");


      // Clean up by closing the source and target files.

      source.close();
      target.close();


      // Get the last changed date/time from the source file and
      // set it on the target file.

      IFSFile src = new IFSFile(as400, sourceName);
      long dateTime = src.lastModified();

      IFSFile tgt = new IFSFile(as400, targetName);
      tgt.setLastModified(dateTime);

      System.out.println("Date/Time successfully set on target file");
      System.out.println("Copy Successful");

   }
   catch (Exception e)
   {
      // If any of the above operations failed say the copy failed
      // and output the exception.

      System.out.println("Copy failed");
      System.out.println(e);
   }
}

// Display help text when parameters are incorrect.

else
{
   System.out.println("");
   System.out.println("");
```

```
         System.out.println("");
         System.out.println("Parameters are not correct.  Command syntax is:");
         System.out.println("");
         System.out.println("   IFSCopyFile as400 source target");
         System.out.println("");
         System.out.println("Where");
         System.out.println("");
         System.out.println("   as400  = system that contains the files");
         System.out.println("   source = source file in /path/path/name format");
         System.out.println("   target = target file in /path/path/name format");
         System.out.println("");
         System.out.println("For example:");
         System.out.println("");
         System.out.println("   IFSCopyFile myAS400  /dir1/dir2/a.txt  /dir3/b.txt");
         System.out.println("");
         System.out.println("");
      }

      System.exit(0);
   }
}
```

範例：使用 IFS 範例列出目錄內容

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////
//
// IFSListFile example.  This program uses the integrated file system classes
// to list the contents of a directory on the server.
//
// Command syntax:
//    IFSList system directory
//
// For example,
//    IFSList MySystem /path1
//
////////////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class IFSList extends Object
{
   public static void main(String[] parameters)
   {
            System.out.println( " " );

            String directoryName = "";
            String system        = "";


              // if both parameters were not specified, display help text and exit.

      if (parameters.length >= 2)
      {

                  // Assume the first parameter is the system name and
         // the second parameter is the directory name

         system = parameters[0];
         directoryName = parameters[1];

         try
         {
            // Create an AS400 object for the server that holds the files.
```

```
            AS400 as400 = new AS400(system);



            // Create the IFSFile object for the directory.

            IFSFile directory = new IFSFile(as400, directoryName);



            // Generate the list of name.  Pass the list method the
            // directory filter object and the search match criteria.
            //
            // Note - this example does the processing in the filter
            // object.  An alternative is to process the list after
            // it is returned from the list method call.

            String[] directoryNames = directory.list(new MyDirectoryFilter(),"*");



            // Tell the user if the directory doesn't exist or is empty

            if (directoryNames == null)
               System.out.println("The directory does not exist");

            else if (directoryNames.length == 0)
               System.out.println("The directory is empty");
         }

         catch (Exception e)
         {
            // If any of the above operations failed say the list failed
            // and output the exception.

            System.out.println("List failed");
            System.out.println(e);
         }
      }

      // Display help text when parameters are incorrect.

      else
      {
         System.out.println("");
         System.out.println("");
         System.out.println("");
         System.out.println("Parameters are not correct.  Command syntax is:");
         System.out.println("");
         System.out.println("   IFSList as400 directory");
         System.out.println("");
         System.out.println("Where");
         System.out.println("");
         System.out.println("   as400 = system that contains the files");
         System.out.println("   directory = directory to be listed");
         System.out.println("");
         System.out.println("For example:");
         System.out.println("");
         System.out.println("   IFSCopyFile mySystem /dir1/dir2");
         System.out.println("");
         System.out.println("");
      }

      System.exit(0);
   }
}
```

```
////////////////////////////////////////////////////////////////////////////
//
// The directory filter class prints information from the file object.
//
// Another way to use the filter is to simply return true or false
// based on information in the file object.  This lets the mainline
// function decide what to do with the list of files that meet the
// search criteria.
//
////////////////////////////////////////////////////////////////////////////



class MyDirectoryFilter implements IFSFileFilter
{
   public boolean accept(IFSFile file)
   {
      try
      {
         // Print the name of the current file

         System.out.print(file.getName());



         // Pad the output so the columns line up

         for (int i = file.getName().length(); i < 18; i++)
            System.out.print(" ");



         // Print the date the file was last changed.

         long changeDate = file.lastModified();
         Date d = new Date(changeDate);
         System.out.print(d);
         System.out.print("  ");



         // Print if the entry is a file or directory

         System.out.print("   ");

         if (file.isDirectory())
            System.out.println("<DIR>");
         else
            System.out.println(file.length());



         // Keep this entry.  Returning true tells the IFSList object
         // to return this file in the list of entries returned to the
         // .list() method.

         return true;
      }

      catch (Exception e)
      {
         return false;
      }
   }
}
```

範例：使用 JDBCPopulate 以建立及輸入表格資料

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////
//
// JDBCPopulate example.This program uses the IBM Toolbox for Java JDBC driver to
// create and populate a table.
//
// Command syntax:
//JDBCPopulate system collectionName tableName
//
// For example,
//JDBCPopulate MySystem MyLibrary MyTable
//
////////////////////////////////////////////////////////////////////////////

import java.sql.*;

public class JDBCPopulate
{



    // Strings to be added in the WORD column of the table.
    private static final String words[]
        = { "One",      "Two",      "Three",    "Four",     "Five",
            "Six",      "Seven",    "Eight",    "Nine",     "Ten",
            "Eleven",   "Twelve",   "Thirteen", "Fourteen", "Fifteen",
            "Sixteen",  "Seventeen","Eighteen", "Nineteen", "Twenty" };



    public static void main(String[] parameters)
    {
        // Check the input parameters.
        if (parameters.length != 3) {
            System.out.println("");
            System.out.println("Usage:");
            System.out.println("");
            System.out.println("   JDBCPopulate system collectionName tableName");
            System.out.println("");
            System.out.println("");
            System.out.println("For example:");
            System.out.println("");
            System.out.println("");
            System.out.println("   JDBCPopulate MySystem MyLibrary MyTable");
            System.out.println("");
            return;
        }

        String system = parameters[0];
        String collectionName = parameters[1];
        String tableName= parameters[2];

        Connection connection = null;

        try {

            // Load the IBM Toolbox for Java JDBC driver.

DriverManager.registerDriver (new com.ibm.as400.access.AS400JDBCDriver
());

            // Get a connection to the database.Since we do not
// provide a user id or password, a prompt will appear.
            //
```

```
            // Note that we provide a default schema here so
            // that we do not need to qualify the table name in
            // SQL statements.
            //
            connection = DriverManager.getConnection ("jdbc:as400://" + system + "/" + collectionName);

            // Drop the table if it already exists.
            try {
                Statement dropTable = connection.createStatement ();
                dropTable.executeUpdate ("DROP TABLE " + tableName);
            }
            catch (SQLException e) {
                // Ignore.
            }

            // Create the table.
            Statement createTable = connection.createStatement ();
            createTable.executeUpdate ("CREATE TABLE " + tableName
                + " (I INTEGER, WORD VARCHAR(20), SQUARE INTEGER, "
                + " SQUAREROOT DOUBLE)");

            // Prepare a statement for inserting rows.Since we
// execute this multiple times, it is best to use a
            // PreparedStatement and parameter markers.
            PreparedStatement insert = connection.prepareStatement ("INSERT INTO "
                + tableName + " (I, WORD, SQUARE, SQUAREROOT) " + " VALUES (?, ?, ?, ?)");

            // Populate the table.
            for (int i = 1; i <= words.length; ++i) {
                insert.setInt (1, i);
                insert.setString (2, words[i-1]);
                insert.setInt (3, i*i);
                insert.setDouble (4, Math.sqrt(i));
                insert.executeUpdate ();
            }

            // Output a completion message.
            System.out.println ("Table " + collectionName + "." + tableName + " has been populated.");
        }

        catch (Exception e) {
            System.out.println ();
            System.out.println ("ERROR: " + e.getMessage());
        }

        finally {

            // Clean up.
            try {
                if (connection != null)
                    connection.close ();
            }
            catch (SQLException e) {
                // Ignore.
            }
        }

        System.exit (0);
    }


}
```

範例：使用 JDBCQuery 以查詢表格

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
///////////////////////////////////////////////////////////////////////////////
//
// JDBCQuery example.This program uses the IBM Toolbox for Java JDBC driver to
// query a table and output its contents.
//
// Command syntax:
//JDBCQuery system collectionName tableName
//
// For example,
//JDBCQuery MySystem qiws qcustcdt
//
///////////////////////////////////////////////////////////////////////////////

import java.sql.*;

public class JDBCQuery
{



    // Format a string so that it has the specified width.
    private static String format (String s, int width)
    {
        String formattedString;

        // The string is shorter than specified width,
        // so we need to pad with blanks.
        if (s.length() < width) {
            StringBuffer buffer = new StringBuffer (s);
            for (int i = s.length(); i < width; ++i)
                buffer.append (" ");
            formattedString = buffer.toString();
        }

        // Otherwise, we need to truncate the string.
        else
            formattedString = s.substring (0, width);

        return formattedString;
    }



    public static void main(String[] parameters)
    {
        // Check the input parameters.
        if (parameters.length != 3) {
            System.out.println("");
            System.out.println("Usage:");
            System.out.println("");
            System.out.println("   JDBCQuery system collectionName tableName");
            System.out.println("");
            System.out.println("");
            System.out.println("For example:");
                        System.out.println("");
            System.out.println("");
            System.out.println("   JDBCQuery mySystem qiws qcustcdt");
            System.out.println("");
            return;
        }

        String system = parameters[0];
        String collectionName = parameters[1];
```

```
        String tableName= parameters[2];

        Connection connection = null;

        try {

            // Load the IBM Toolbox for Java JDBC driver.

DriverManager.registerDriver (new com.ibm.as400.access.AS400JDBCDriver
());

            // Get a connection to the database.Since we do not
// provide a user id or password, a prompt will appear.
            connection = DriverManager.getConnection ("jdbc:as400://" + system);
            DatabaseMetaData dmd = connection.getMetaData ();

            // Execute the query.
            Statement select = connection.createStatement ();
            ResultSet rs = select.executeQuery (
               "SELECT * FROM " + collectionName + dmd.getCatalogSeparator() + tableName);

            // Get information about the result set.Set the column
// width to whichever is longer: the length of the label
            // or the length of the data.
            ResultSetMetaData rsmd = rs.getMetaData ();
            int columnCount = rsmd.getColumnCount ();
            String[] columnLabels = new String[columnCount];
            int[] columnWidths = new int[columnCount];
            for (int i = 1; i <= columnCount; ++i) {
                columnLabels[i-1] = rsmd.getColumnLabel (i);
                columnWidths[i-1] = Math.max (columnLabels[i-1].length(), rsmd.getColumnDisplaySize (i));
            }

            // Output the column headings.
            for (int i = 1; i <= columnCount; ++i) {
                System.out.print (format (rsmd.getColumnLabel(i), columnWidths[i-1]));
                                  System.out.print("  ");
            }
            System.out.println ();

            // Output a dashed line.
            StringBuffer dashedLine;
            for (int i = 1; i <= columnCount; ++i) {
                for (int j = 1; j <= columnWidths[i-1]; ++j)
                    System.out.print ("-");
                System.out.print (" ");
            }
            System.out.println ();

            // Iterate throught the rows in the result set and output
            // the columns for each row.
            while (rs.next ()) {
                for (int i = 1; i <= columnCount; ++i) {
                    String value = rs.getString (i);
                    if (rs.wasNull ())
                        value = "<null>";
                    System.out.print (format (value, columnWidths[i-1]));
                    System.out.print (" ");
                }
                System.out.println ();
            }

        }

        catch (Exception e) {
            System.out.println ();
            System.out.println ("ERROR: " + e.getMessage());
```

```
        }

        finally {

            // Clean up.
            try {
                if (connection != null)
                    connection.close ();
            }
            catch (SQLException e) {
                // Ignore.
            }
        }

        System.exit (0);
    }



}
```

範例：使用 JobList 以列出工作識別資訊

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////
//
// This program is an example of the Job support in the IBM Toolbox
// for Java.It lists job identification information for a specific
// user on the system.
//
// Command syntax:
//listJobs2 system userID password
//
////////////////////////////////////////////////////////////////////////

import java.io.*;
import java.lang.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class listJobs2 extends Object
{

                    // Create an object in case we want to call
                    // any non-staic methods.
   public static void main(String[] parameters)
   {
      listJobs2 me = new listJobs2();
      me.Main(parameters);

      System.exit(0);
   }



   void Main(String[] parameters)
   {

                    // If a system was not specified, display help text and exit.
      if (parameters.length == 0)
      {
         showHelp();
         return;
      }

                    // Assign the parameters to variables.The
```

```
// first parameter is assumed to be the system
                    // name the second is a userID and the third
                    // is a password.
      String systemName = parameters[0];
      String userID = null;
      String password = null;

      if (parameters.length > 1)
         userID = parameters[1].toUpperCase();

      if (parameters.length >= 2)
         password = parameters[2].toUpperCase();

      System.out.println(" ");



      try
      {

                    // Create an AS400 object using the system name
                    // specified by the user. Set the userid and
// password if specified by the user.
         AS400 as400 = new AS400(parameters[0]);

         if (userID != null)
            as400.setUserId(userID);

         if (password != null)
            as400.setPassword(password);



         System.out.println("retrieving list ...     ");


                    // Create a jobList object.This object is used
// to retrieve the list of active jobs on the server.
         JobList jobList = new JobList(as400);


                    // Get the list of active jobs.
         Enumeration list = jobList.getJobs();


                    // For each job in the list ...
         while (list.hasMoreElements())
         {

                    // Get a job off the list.If a userID was
// specified then print identification information
                    // only if the job's user matches the userID.If
// no userID was specified then print information
                    // for every job on the system.
            Job j = (Job) list.nextElement();

            if (userID != null)
            {
               if (j.getUser().trim().equalsIgnoreCase(userID))
               {
                  System.out.println(j.getName().trim() + "." +
                                     j.getUser().trim() + "." +
                                     j.getNumber());
               }
            }
            else
               System.out.println(j.getName().trim() + "." +
```

```
                                j.getUser().trim() + "." +
                                j.getNumber());
         }

      }
      catch (Exception e)
      {
         System.out.println("Unexpected error");
         e.printStackTrace();
      }
   }



      // Display help text when parameters are incorrect.
   void showHelp()
   {
         System.out.println("");
         System.out.println("");
         System.out.println("");
         System.out.println("Parameters are not correct.  Command syntax is:");
         System.out.println("");
         System.out.println("   listJobs2 System UserID Password");
         System.out.println("");
         System.out.println("Where");
         System.out.println("");
         System.out.println("   System   = server to connect to");
         System.out.println("   UserID   = valid userID on that system ");
         System.out.println("   Password = password for the UserID (optional)");
         System.out.println("");
         System.out.println("For example:");
         System.out.println("");
         System.out.println("   listJobs2 MYAS400 JavaUser pwd1");
         System.out.println("");
         System.out.println("");
   }
}
```

範例：使用 JobList 以取得工作清單

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////
//
// This program is an example of the "job" classes in the
// IBM Toolbox for Java.  It gets a list of jobs on the server
// and outputs the job's status followed by job identifier.
//
//
// Command syntax:
//listJobs system userID password
//
// (UserID and password are optional)
//
////////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class listJobs extends Object
{
   public static void main(String[] parameters)
   {
      listJobs me = new listJobs();
      me.Main(parameters);
```

```
            System.exit(0);
   }



    void Main(String[] parameters)
   {

      // If a system was not specified, display help text and exit.
      if (parameters.length == 0)
      {
          showHelp();
             return;
      }

      // Set up AS400 object parms.The first is the system name and must
// be specified by the user.The second and third are optional.They
// are the userid and password.Convert the userid and password
// to uppercase before setting them on the AS400 object.
      String userID = null;
      String password = null;

      if (parameters.length > 1)
          userID = parameters[1].toUpperCase();

      if (parameters.length >= 2)
          password = parameters[2].toUpperCase();

      System.out.println(" ");

       try
      {

          // Create an AS400 object using the system name specified by the user.
         AS400 as400 = new AS400(parameters[0]);

          // If a userid and/or password was specified, set them on the
          // AS400 object.
         if (userID != null)
            as400.setUserId(userID);

         if (password != null)
            as400.setPassword(password);

         // Create a job list object.  Input parm is the AS400 we want job
         // information from.
         JobList jobList = new JobList(as400);

         // Get a list of jobs running on the server.
         Enumeration listOfJobs = jobList.getJobs();

         // For each job in the list print information about the job.
         while (listOfJobs.hasMoreElements())
         {
            printJobInfo((Job) listOfJobs.nextElement(), as400);
         }

      }
      catch (Exception e)
      {
         System.out.println("Unexpected error");
         System.out.println(e);
      }
   }
```

```
   void printJobInfo(Job job, AS400 as400)
   {

      // Create the various converters we need
      AS400Bin4 bin4Converter   = new AS400Bin4( );
      AS400Text text26Converter = new AS400Text(26, as400);
      AS400Text text16Converter = new AS400Text(16, as400);
      AS400Text text10Converter = new AS400Text(10, as400);
      AS400Text text8Converter= new AS400Text(8,as400);
      AS400Text text6Converter= new AS400Text(6,as400);
      AS400Text text4Converter= new AS400Text(4,as400);


      // We have the job name/number/etc. from the list request.Now
// make a server API call to get the status of the job.
      try
      {
         // Create a program call object
         ProgramCall pgm = new ProgramCall(as400);

         // The server program we call has five parameters
         ProgramParameter[] parmlist = new ProgramParameter[5];

         // The first parm is a byte array that holds the output
         // data.  We will allocate a 1k buffer for output data.
         parmlist[0] = new ProgramParameter( 1024 );

         // The second parm is the size of our output data buffer (1K).
         Integer iStatusLength = new Integer( 1024 );
         byte[]  statusLength = bin4Converter.toBytes( iStatusLength );
         parmlist[1] = new ProgramParameter( statusLength );

         // The third parm is the name of the format of the data.
         // We will use format JOBI0200 because it has job status.
         byte[] statusFormat = text8Converter.toBytes("JOBI0200");
         parmlist[2] = new ProgramParameter( statusFormat );

         // The fourth parm is the job name is format "name user number".
         // Name must be 10 characters, user must be 10 characters and
         // number must be 6 characters.  We will use a text converter
         // to do the conversion and padding.
         byte[] jobName = text26Converter.toBytes(job.getName());

         int    i       = text10Converter.toBytes(job.getUser(),
                                                  jobName,
                                                  10);

                i = text6Converter.toBytes(job.getNumber(),
                                           jobName,
                                           20);

         parmlist[3] = new ProgramParameter( jobName );

         // The last paramter is job identifier.  We will leave this blank.
         byte[] jobID = text16Converter.toBytes("                ");
         parmlist[4] = new ProgramParameter( jobID );


         // Run the program.
         if (pgm.run( "/QSYS.LIB/QUSRJOBI.PGM", parmlist )==false)
         {
            // if the program failed display the error message.
            AS400Message[] msgList = pgm.getMessageList();
            System.out.println(msgList[0].getText());
         }
         else
         {
```

```
            // else the program worked.Output the status followed by
// the jobName.user.jobID
            byte[] as400Data = parmlist[0].getOutputData();
            System.out.print("  " + text4Converter.toObject(as400Data, 107) + "  ");

            System.out.println(job.getName().trim() + "." +
                               job.getUser().trim() + "." +
                               job.getNumber() + "   ");
         }

      }
      catch (Exception e)
      {
         System.out.println(e);
      }

   }



      // Display help text when parameters are incorrect.
   void showHelp()
   {
         System.out.println("");
         System.out.println("");
         System.out.println("");
         System.out.println("Parameters are not correct.  Command syntax is:");
         System.out.println("");
         System.out.println("   listJobs System UserID Password");
         System.out.println("");
         System.out.println("Where");
         System.out.println("");
         System.out.println("   System   = server to connect to");
         System.out.println("   UserID   = valid userID on that system (optional)");
         System.out.println("   Password = password for the UserID (optional)");
         System.out.println("");
         System.out.println("For example:");
         System.out.println("");
         System.out.println("   listJobs MYAS400 JavaUser pwd1");
         System.out.println("");
         System.out.println("");
   }
}
```

範例：使用 JobLog 以顯示工作日誌中的訊息

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
///////////////////////////////////////////////////////////////////////////
//
// This program is an example of the job log fuction of the
// IBM Toolbox for Java.  It will display the messages in the job
// log for a job that belongs to the current user.
//
// Command syntax:
//jobLogExample system userID password
//
// (Password is optional)
//
///////////////////////////////////////////////////////////////////////////

import java.lang.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class jobLogExample
{
```

```
     public static void main(String[] args)
    {
        // If a system and user were not specified, display help text and exit.
        if (args.length < 2)
        {
            System.out.println("Usage:  jobLogExample system userid <password>");
            return;
        }

        String userID = null;

        try
        {
            // Create an AS400 object.  The system name was passed
            // as the first command line argument.
If a userid
            // and password were passed on the command line,
            // set those as well.
            AS400 system = new AS400 (args[0]);

            if (args.length > 1)
            {
               userID = args[1];
               system.setUserId(userID);
            }

            if (args.length > 2)
               system.setPassword(args[2]);


            // Create a job list object.This object will be used to get
// the list of active jobs on the system.Once the list of
// jobs is retrieved, the program will find a job for the
            // current user.
            JobList jobList = new JobList(system);

            // Get the list of active jobs on the AS/400
            Enumeration list = jobList.getJobs();

            boolean Continue = true;

            // Look through the list to find a job for the current user.
            while (list.hasMoreElements() && Continue)
            {
               Job j = (Job) list.nextElement();

               if (j.getUser().trim().equalsIgnoreCase(userID))
               {
                  // A job matching the current user was found.  Create
                  // a job log object for this job.
                  JobLog jlog = new JobLog(system, j.getName(), j.getUser(), j.getNumber());

                  // Enumerate the messages in the job log then print them.
                  Enumeration messageList = jlog.getMessages();

                  while (messageList.hasMoreElements())
                  {
                     AS400Message message = (AS400Message) messageList.nextElement();
                     System.out.println(message.getText());
                  }

                  // We found one job matching the current user so exit.
                  Continue = false;
               }
            }
```

```
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
        }

        System.exit(0);
    }

}
```

範例：建立排存檔

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// Example that shows creating a spooled file on a server from an input stream.
//
///////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;

import com.ibm.as400.access.*;

class NPExampleCreateSplf
{

// method to create the spooled file on the specified server, in the specified
// output queue from the given input stream.
public SpooledFile createSpooledFile(AS400 system, OutputQueue outputQueue, InputStream in)
{
    SpooledFile spooledFile = null;
     try
    {
        byte[] buf = new byte[2048];
        int bytesRead;
        SpooledFileOutputStream out;
        PrintParameterList parms = new PrintParameterList();

        // create a PrintParameterList with the values that we want
        // to override from the default printer file...we will override
        // the output queue and the copies value.
        parms.setParameter(PrintObject.ATTR_COPIES, 4);
        if (outputQueue != null)
        {
           parms.setParameter(PrintObject.ATTR_OUTPUT_QUEUE, outputQueue.getPath());
        }
        out = new SpooledFileOutputStream(system,
                                          parms,
                                          null,
                                          null);

        // read from the inputstream in until end of stream, passing all data
        // to the spooled file output stream.
        do
        {
            bytesRead = in.read(buf);
            if (bytesRead != -1)
            {
                out.write(buf, 0, bytesRead);
            }
        } while (bytesRead != -1);

        out.close();// close the spooled file
```

```
        spooledFile = out.getSpooledFile(); // get a reference to the new spooled file

    }
    catch (Exception e)
    {
        //...handle exception...
    }
    return spooledFile;
}

}
```

範例：建立 SCS 排存檔

此範例使用 SCS3812Writer 類別來產生 SCS 資料串流，並將它寫入伺服器上的排存檔。

此應用程式可以採用下列引數，或可以使用已定義的預設值：

- 接收排存檔的伺服器名稱。
- 伺服器上接收排存檔的輸出佇登記稱。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// This source is an example of IBM Toolbox for Java "SCS3812Writer".
//
///////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;

class NPExampleCreateSCSSplf
{
    private static final String DEFAULT_SYSTEM = new String("RCHAS1");
    private static final String DEFAULT_OUTQ = new String("/QSYS.LIB/QUSRSYS.LIB/PRT01.OUTQ");

     public static void main(String[] args)
    {
         try
        {
            AS400 system;
            SpooledFileOutputStream out;
            PrintParameterList parms = new PrintParameterList();
            SCS3812Writer scsWtr;

            // Process the arguments.
            if (args.length >= 1)
            {
                system = new AS400(args[0]);// Create an AS400 object
            } else {
                system = new AS400(DEFAULT_SYSTEM);
            }

            if (args.length >= 2)               // Set the outq
            {
                parms.setParameter(PrintObject.ATTR_OUTPUT_QUEUE, args[1]);
            } else {
                parms.setParameter(PrintObject.ATTR_OUTPUT_QUEUE, DEFAULT_OUTQ);
            }

            out = new SpooledFileOutputStream(system, parms, null, null);

            scsWtr = new SCS3812Writer(out, 37);

            // Write the contents of the spool file.
```

```
            scsWtr.setLeftMargin(1.0);
            scsWtr.absoluteVerticalPosition(6);
            scsWtr.setFont(scsWtr.FONT_COURIER_BOLD_5);
            scsWtr.write("          Java Printing");
            scsWtr.newLine();
            scsWtr.newLine();
            scsWtr.setCPI(10);
            scsWtr.write("This document was created using the IBM Toolbox for Java.");
            scsWtr.newLine();
            scsWtr.write("The rest of this document shows some of the things that");
            scsWtr.newLine();
            scsWtr.write("can be done with the SCS3812Writer class.");
            scsWtr.newLine();
            scsWtr.newLine();
            scsWtr.setUnderline(true); scsWtr.write("Setting fonts:"); scsWtr.setUnderline(false);
            scsWtr.newLine();
            scsWtr.setFont(scsWtr.FONT_COURIER_10); scsWtr.write("Courier font ");
            scsWtr.setFont(scsWtr.FONT_COURIER_BOLD_10); scsWtr.write(" Courier bold font ");
            scsWtr.setFont(scsWtr.FONT_COURIER_ITALIC_10); scsWtr.write(" Courier italic font ");
            scsWtr.newLine();
            scsWtr.setBold(true); scsWtr.write("Courier bold italic font ");
            scsWtr.setBold(false);
            scsWtr.setCPI(10);
            scsWtr.newLine();
            scsWtr.newLine();
            scsWtr.setUnderline(true); scsWtr.write("Lines per inch:"); scsWtr.setUnderline(false);
            scsWtr.newLine();
            scsWtr.write("The following lines should print at 8 lines per inch.");
            scsWtr.newLine();
            scsWtr.newLine();
            scsWtr.setLPI(8);
            scsWtr.write("Line one"); scsWtr.newLine();
            scsWtr.write("Line two"); scsWtr.newLine();
            scsWtr.write("Line three"); scsWtr.newLine();
            scsWtr.write("Line four"); scsWtr.newLine();
            scsWtr.write("Line five"); scsWtr.newLine();
            scsWtr.write("Line six"); scsWtr.newLine();
            scsWtr.write("Line seven"); scsWtr.newLine();
            scsWtr.write("Line eight"); scsWtr.newLine();
            scsWtr.endPage();
            scsWtr.setLPI(6);
            scsWtr.setSourceDrawer(1);
            scsWtr.setTextOrientation(0);
            scsWtr.absoluteVerticalPosition(6);
            scsWtr.write("This page should print in portrait orientation from drawer 1.");
            scsWtr.endPage();
            scsWtr.setSourceDrawer(2);
            scsWtr.setTextOrientation(90);
            scsWtr.absoluteVerticalPosition(6);
            scsWtr.write("This page should print in landscape orientation from drawer 2.");
            scsWtr.endPage();
            scsWtr.close();
            System.out.println("Sample spool file created.");
            System.exit(0);
        }
        catch (Exception e)
        {
            // Handle error.
            System.out.println("Exception occured while creating spooled file. " + e);
            System.exit(0);
        }
    }
}
```

範例：讀取排存檔

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// Example that reads an existing server spooled file.
//
// This source is an example of IBM Toolbox for Java "PrintObjectInputStream".
//
////////////////////////////////////////////////////////////////////////
        try{
        byte[] buf = new byte[2048];
        int bytesRead;
        AS400 sys = new AS400();
        SpooledFile splf = new SpooledFile( sys,// AS400
                                            "MICR",       // splf name
                                            17, // splf number
                                            "QPRTJOB",    // job name
                                            "QUSER",      // job user
                                            "020791" );   // job number

        // open the spooled file for reading and get the input stream to
        // read from it.
        InputStream in = splf.getInputStream(null);

        do
        {
            // read up to buf.length bytes of raw spool data into
            // our buffer.The actual bytes read will be returned.
// The data will be a binary printer data stream that is the
            // contents of the spooled file.
            bytesRead = in.read( buf );
            if( bytesRead != -1 )
            {
                // process the spooled file data.
               System.out.println( "Read " + bytesRead + " bytes" );
            }
        } while( bytesRead != -1 );

        in.close();
    }
    catch (Exception e)
    {
        // exception
    }
```

範例：讀取與轉換排存檔

下列範例示範如何設定 PrintParameterList，以在讀取排存檔資料時獲得不同的轉換。在接下來的程式區段中，假設排存檔已在伺服器上，且 *createSpooledFile()* 方法會建立代表排存檔之 SpooledFile 類別的案例。

PrintObjectPageInputStream 範例

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

下列範例顯示針對讀取格式化爲 GIF 影像的資料頁面時，應如何建立 PrintObjectPageInputStream 物件。 在此情況下，來自排存檔的每一頁都將轉換爲 GIF 影像。GIF 工作站自訂物件用來指定資料轉換。

```
// Create a spooled file
SpooledFile splF = createSpooledFile();

// Set up print parameter list
PrintParameterList printParms = new PrintParameterList();
```

```
printParms.setParameter(PrintObject.ATTR_WORKSTATION_CUST_OBJECT, "/QSYS.LIB/QWPGIF.WSCST");
printParms.setParameter(PrintObject.ATTR_MFGTYPE, "*WSCST");

// Create a page input stream from the spooled file
PrintObjectPageInputStream is = splF.getPageInputStream(printParms);
```

PrintObjectTransformedInputStream 範例

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

下列範例顯示針對讀取格式化爲 TIFF 的資料時，應如何建立 PrintObjectTransformedInputStream 物件。TIFF (G4 壓縮) 工作站自訂物件用來指定資料轉換。

```
// Create a spooled file
SpooledFile splF = createSpooledFile();

// Set up print parameter list
PrintParameterList printParms = new PrintParameterList();
printParms.setParameter(PrintObject.ATTR_WORKSTATION_CUST_OBJECT, "/QSYS.LIB/QWPTIFFG4.WSCST");
printParms.setParameter(PrintObject.ATTR_MFGTYPE, "*WSCST");

// Create a transformed input stream from the spooled file
PrintObjectTransformedInputStream is = splF.getTransformedInputStream(printParms);
```

使用製造商類型及模型的 PrintObjectTransformedInputStream 範例

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

下列範例針對爲輸出到 ASCII 印表機所格式化的資料，說明如何建立讀取此類資料的 PrintObjectTransformedInputStream 物件。製造商類型及 *HP4 模型用來指定資料轉換。

```
// Create a spooled file
SpooledFile splF = createSpooledFile();

// Set up print parameter list
PrintParameterList printParms = new PrintParameterList();
printParms.setParameter(PrintObject.ATTR_MFGTYPE, "*HP4");

// Create a transformed input stream from the spooled file
PrintObjectTransformedInputStream is = splF.getTransformedInputStream(printParms);
```

範例：非同步列出排存檔 (使用接收程式)

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// Example that shows listing all spooled files on a server asynchronously using
// the PrintObjectListListener interface to get feedback as the list is being built.
// Listing asynchronously allows the caller to start processing the list objects
// before the entire list is built for a faster perceived response time
// for the user.
//
///////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.SpooledFileList;
import com.ibm.as400.access.SpooledFile;
import com.ibm.as400.access.ExtendedIllegalStateException;
import com.ibm.as400.access.PrintObjectListListener;
import com.ibm.as400.access.PrintObjectListEvent;

public class NPExampleListSplfAsynch extends Object implements PrintObjectListListener
{
```

```
private AS400 system_;
private boolean fListError;
private boolean fListClosed;
private boolean fListCompleted;
private Exception listException;
private int listObjectCount;

public NPExampleListSplfAsynch(AS400 system)
{
    system_ = system;
}

// list all spooled files on the server asynchronously using a listener
public void listSpooledFiles()
{
    fListError = false;
    fListClosed = false;
    fListCompleted = false;
    listException = null;
    listObjectCount = 0;

     try
    {
        String strSpooledFileName;
        boolean fCompleted = false;
        int listed = 0, size;

        if (system_ == null)
        {
            system_ = new AS400();
        }

        System.out.println(" Now receiving all spooled files Asynchronously using a listener");

        SpooledFileList splfList = new SpooledFileList(system_);

        // set filters, all users, on all queues
        splfList.setUserFilter("*ALL");
        splfList.setQueueFilter("/QSYS.LIB/%ALL%.LIB/%ALL%.OUTQ");

        // add the listener.
        splfList.addPrintObjectListListener(this);

        // open the list, openAsynchronously returns immediately
        splfList.openAsynchronously();

        do
        {
            // wait for the list to have at least 25 objects or to be done
            waitForWakeUp();

            fCompleted = splfList.isCompleted();
            size = splfList.size();

            // output the names of all objects added to the list
            // since we last woke up
            while (listed < size)
            {
                if (fListError)
                {
                    System.out.println(" Exception on list - " + listException);
                    break;
                }

                if (fListClosed)
                {
                    System.out.println(" The list was closed before it completed!");
```

```
                    break;
                }

                SpooledFile splf = (SpooledFile)splfList.getObject(listed++);
                if (splf != null)
                {
                    // output this spooled file name
                    strSpooledFileName = splf.getStringAttribute(SpooledFile.ATTR_SPOOLFILE);
                    System.out.println(" spooled file = " + strSpooledFileName);
                }
            }

        } while (!fCompleted);

        // clean up after we are done with the list
        splfList.close();
        splfList.removePrintObjectListListener(this);
    }

    catch( ExtendedIllegalStateException e )
    {
        System.out.println(" The list was closed before it completed!");
    }

    catch (Exception e)
    {
        // ...handle any other exceptions...
        e.printStackTrace();
    }

}

// This is where the foreground thread waits to be awaken by the
// the background thread when the list is updated or it ends.
private synchronized void waitForWakeUp() throws InterruptedException
{
    // don''t go back to sleep if the listener says the list is done
    if (!fListCompleted)
    {
        wait();
    }
}

// The following methods implement the PrintObjectListListener interface

// This method is invoked when the list is closed.
public void listClosed(PrintObjectListEvent event)
{
    System.out.println("*****The list was closed*****");
    fListClosed = true;
    synchronized(this)
    {
        // Set flag to indicate that the list has
        // completed and wake up foreground thread.
        fListCompleted = true;
        notifyAll();
    }
}

// This method is invoked when the list is completed.
public void listCompleted(PrintObjectListEvent event)
{
    System.out.println("*****The list has completed*****");
    synchronized (this)
    {
        // Set flag to indicate that the list has
        // completed and wake up foreground thread.
```

```
            fListCompleted = true;
            notifyAll();
        }
    }

    // This method is invoked if an error occurs while retrieving
    // the list.
    public void listErrorOccurred(PrintObjectListEvent event)
    {
        System.out.println("*****The list had an error*****");
        fListError = true;
        listException = event.getException();
        synchronized(this)
        {
            // Set flag to indicate that the list has
            // completed and wake up foreground thread.
            fListCompleted = true;
            notifyAll();
        }
    }

    // This method is invoked when the list is opened.
    public void listOpened(PrintObjectListEvent event)
    {
        System.out.println("*****The list was opened*****");
        listObjectCount = 0;
    }

    // This method is invoked when an object is added to the list.
    public void listObjectAdded(PrintObjectListEvent event)
    {
        // every 25 objects we'll wake up the foreground
        // thread to get the latest objects...
        if( (++listObjectCount % 25) == 0 )
        {
            System.out.println("*****25 more objects added to the list*****");
            synchronized (this)
            {
                // wake up foreground thread
                notifyAll();
            }
        }
    }

    public static void main( String args[] )
    {
        NPExampleListSplfAsynch list = new NPExampleListSplfAsynch(new AS400());
        try{
            list.listSpooledFiles();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        System.exit(0);
    }

}
```

範例：非同步列出排存檔 (不使用接收程式)

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// Example that shows listing all spooled files on a system Asynchronously without
// using the PrintObjectListListener interface. After opening the list the caller
```

```
// can do some additional work before waiting for the list to complete.
//
////////////////////////////////////////////////////////////////////////
//
// This source is an example of IBM Toolbox for Java "PrintObjectList".
//
////////////////////////////////////////////////////////////////////////

import java.util.Enumeration;

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.SpooledFileList;
import com.ibm.as400.access.SpooledFile;

public class NPExampleListSplfAsynch2 extends Object
{
    private AS400 system_;

    public NPExampleListSplfAsynch2(AS400 system)
    {
        system_ = system;
    }

    // list all spooled files on the system asynchronously
    public void listSpooledFiles()
    {
         try
        {
            String strSpooledFileName;
            int listed, size;

            if (system_ == null)
            {
                system_ = new AS400();
            }

            System.out.println(
               "Now receiving all spooled files Asynchronously without using a listener");

            SpooledFileList splfList = new SpooledFileList(system_);

            // set filters, all users, on all queues
            splfList.setUserFilter("*ALL");
            splfList.setQueueFilter("/QSYS.LIB/%ALL%.LIB/%ALL%.OUTQ");

            // open list, openAsynchronously() returns immediately
            // we have not added any listeners...
            splfList.openAsynchronously();

            System.out.println(" Do some processing before waiting...");

            // ... do some processing here while the list is being built....

            System.out.println(" Now wait for list to complete.");

            // wait for the list to complete
            splfList.waitForListToComplete();

            Enumeration enum = splfList.getObjects();

            // output the name of all objects on the list
            while( enum.hasMoreElements() )
            {
                SpooledFile splf = (SpooledFile)enum.nextElement();
                if (splf != null)
                {
                    // output this spooled file's name
```

```
                    strSpooledFileName = splf.getStringAttribute(SpooledFile.ATTR_SPOOLFILE);
                    System.out.println(" spooled file = " + strSpooledFileName);
                }
            }
            // clean up after we are done with the list
            splfList.close();
        }

        catch (Exception e)
        {
            // ...handle any exceptions...
            e.printStackTrace();
        }
    }

    public static void main( String args[] )
    {
        NPExampleListSplfAsynch2 list = new NPExampleListSplfAsynch2(new AS400());
        try{
            list.listSpooledFiles();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        System.exit(0);
    }
}
```

範例：同步列出排存檔

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////
//
// Example that shows listing all spooled files on a server synchronously.
// Listing synchronously does not return to the caller until the complete list
// is built. The user perceives a slower response time then listing asynchronously.
//
///////////////////////////////////////////////////////////////////////////
//
// This source is an example of IBM Toolbox for Java "PrintObjectList".
//
///////////////////////////////////////////////////////////////////////////

import java.util.Enumeration;

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.SpooledFileList;
import com.ibm.as400.access.SpooledFile;

public class NPExampleListSplfSynch
{
    private AS400 system_ = new AS400();

    public NPExampleListSplfSynch(AS400 system)
    {
        system_ = system;
    }

    public void listSpooledFiles()
    {
        try{
            String strSpooledFileName;

            if (system_ == null)
            {
```

```
            system_ = new AS400();
        }

        System.out.println(" Now receiving all spooled files Synchronously");

        SpooledFileList splfList = new SpooledFileList( system_ );

        // set filters, all users, on all queues
        splfList.setUserFilter("*ALL");
        splfList.setQueueFilter("/QSYS.LIB/%ALL%.LIB/%ALL%.OUTQ");

        // open list, openSynchronously() returns when the list is completed.
        splfList.openSynchronously();
        Enumeration enum = splfList.getObjects();

        while( enum.hasMoreElements() )
        {
            SpooledFile splf = (SpooledFile)enum.nextElement();
            if ( splf != null )
            {
                // output this spooled file's name
                strSpooledFileName = splf.getStringAttribute(SpooledFile.ATTR_SPOOLFILE);
                System.out.println(" spooled file = " + strSpooledFileName);
            }
        }
        // clean up after we are done with the list
        splfList.close();
    }
    catch (Exception e)
    {
        // ...handle any exceptions...
        e.printStackTrace();
    }
}

public static void main( String args[] )
{
    NPExampleListSplfSynch list = new NPExampleListSplfSynch(new AS400());
    try{
        list.listSpooledFiles();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    System.exit(0);
}
}
```

範例：使用 ProgramCall

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// Program call example.This program calls the QWCRSSTS server program
// to retrieve the status of the system.
//
// Command syntax:
//PCSystemStatusExample system
//
// This source is an example of IBM Toolbox for Java "ProgramCall".
//
///////////////////////////////////////////////////////////////////////
```

```
import java.io.*;
 import java.util.*;
import java.math.*;
import java.lang.Thread.*;
import com.ibm.as400.access.*;

public class PCSystemStatusExample extends Object
{
   public static void main(String[] parameters)
   {
            System.out.println( " " );



      // if a system was not specified, display help text and exit.

      if (parameters.length >= 1)
      {

          try
         {
            // Create an AS400 object for the server that contains the
            // program.  Assume the first parameter is the system name.

            AS400 as400 = new AS400(parameters[0]);



            // Create the path to the program.

            QSYSObjectPathName programName = new QSYSObjectPathName("QSYS", "QWCRSSTS", "PGM");



            // Create the program call object.Assocate the object with the
// AS400 object that represents the server we get status from.

            ProgramCall getSystemStatus = new ProgramCall(as400);



            // Create the program parameter list.This program has five
// parameters that will be added to this list.

            ProgramParameter[] parmlist = new ProgramParameter[5];



            // The server program returns data in parameter 1.It is an output
// parameter.  Allocate 64 bytes for this parameter.

            parmlist[0] = new ProgramParameter( 64 );



            // Parameter 2 is the buffer size of parm 1.It is a numeric input
// parameter.  Sets its value to 64, convert it to the server format,
            // then add the parm to the parm list.

            AS400Bin4 bin4 = new AS400Bin4( );
            Integer iStatusLength = new Integer( 64 );
            byte[]statusLength = bin4.toBytes( iStatusLength );
            parmlist[1] = new ProgramParameter( statusLength );
```

```
            // Parameter 3 is the status-format parameter.It is a string input
// parameter.  Set the string value, convert it to the server format,
            // then add the parameter to the parm list.

            AS400Text text1 = new AS400Text(8, as400);
            byte[] statusFormat = text1.toBytes("SSTS0200");
            parmlist[2] = new ProgramParameter( statusFormat );



            // Parameter 4 is the reset-statistics parameter.It is a string input
// parameter.  Set the string value, convert it to the server format,
            // then add the parameter to the parm list.

            AS400Text text3 = new AS400Text(10, as400);
            byte[] resetStats = text3.toBytes("*NO       ");
            parmlist[3] = new ProgramParameter( resetStats );



            // Parameter 5 is the error info parameter.It is an input/output
// parameter.  Add it to the parm list.

            byte[] errorInfo = new byte[32];
            parmlist[4] = new ProgramParameter( errorInfo, 0 );



            // Set the program to call and the parameter list to the program
            // call object.

            getSystemStatus.setProgram(programName.getPath(), parmlist );



            // Run the program then sleep.We run the program twice because
// the first set of results are inflated.If we discard the first
// set of results and run the command again five seconds later the
            // number will be more accurate.

            getSystemStatus.run();
            Thread.sleep(5000);



            // Run the program

            if (getSystemStatus.run()!=true)
            {

               // If the program did not run get the list of error messages
               // from the program object and display the messages.  The error
               // would be something like program-not-found or not-authorized
               // to the program.

               AS400Message[] msgList = getSystemStatus.getMessageList();

               System.out.println("The program did not run.  Server messages:");

               for (int i=0; i<msgList.length; i++)
               {
                   System.out.println(msgList[i].getText());
               }
            }


            // Else the program did run.
```

```
   else
   {

      // Create a server to Java numeric converter.  This converter
      // will be used in the following section to convert the numeric
      // output from the server format to Java format.

      AS400Bin4 as400Int = new AS400Bin4( );



      // Get the results of the program.  Output data is in
      // a byte array in the first parameter.

      byte[] as400Data = parmlist[0].getOutputData();



      // CPU utilization is a numeric field starting at byte
      // 32 of the output buffer.  Convert this number from the
      // server format to Java format and output the number.

      Integer cpuUtil = (Integer)as400Int.toObject( as400Data, 32 );
      cpuUtil = new Integer(cpuUtil.intValue()/10);
      System.out.print("CPU Utilization:  ");
      System.out.print(cpuUtil);
      System.out.println("%");



      // DASD utilization is a numeric field starting at byte
      // 52 of the output buffer.  Convert this number from the
      // server format to Java format and output the number.

      Integer dasdUtil = (Integer)as400Int.toObject( as400Data, 52 );
      dasdUtil = new Integer(dasdUtil.intValue()/10000);
      System.out.print("Dasd Utilization: ");
      System.out.print(dasdUtil);
      System.out.println("%");



      // Number of jobs is a numeric field starting at byte
      // 36 of the output buffer.  Convert this number from the
      // server format to Java format and output the number.

      Integer nj = (Integer)as400Int.toObject( as400Data, 36 );
      System.out.print("Active jobs:      ");
      System.out.println(nj);

   }

      // This program is done running program so disconnect from
      // the command server on the server.  Program call and command
      // call use the same server on the server.

   as400.disconnectService(AS400.COMMAND);
}
catch (Exception e)
{
   // If any of the above operations failed say the program failed
   // and output the exception.

   System.out.println("Program call failed");
   System.out.println(e);
}
```

```
      }

      // Display help text when parameters are incorrect.

      else
      {
         System.out.println("");
         System.out.println("");
         System.out.println("");
         System.out.println("Parameters are not correct.  Command syntax is:");
         System.out.println("");
         System.out.println("   PCSystemStatusExample myServer");
         System.out.println("");
         System.out.println("Where");
         System.out.println("");
         System.out.println("   myServer = get status of this server ");
         System.out.println("");
         System.out.println("For example:");
         System.out.println("");
         System.out.println("   PCSystemStatusExample mySystem");
         System.out.println("");
         System.out.println("");
      }

      System.exit(0);
   }
}
```

範例：使用記錄層次存取類別

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////////
//
// Record level access example.
This program will prompt the user
// for the name of the server and the file to display.The file must exist
// and contain records.  Each record in the file will be displayed
// to System.out.
//
// Calling syntax: java RLSequentialAccessExample
//
// This source is an example of IBM Toolbox for Java "RecordLevelAccess"
//
////////////////////////////////////////////////////////////////////////////////



import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class RLSequentialAccessExample
{
    public static void main(String[] parmeters)
   {
      // Created a reader to get input from the user
      BufferedReader inputStream = new BufferedReader(new InputStreamReader(System.in),1);

      // Declare variables to hold the system name, library, file and member names
      String systemName = "";
      String library = "";
      String file = "";
      String member = "";

      // Get the system name and and file to display from the user
      System.out.println();
```

```
      try
      {
        System.out.print("System name: ");
        systemName = inputStream.readLine();

        System.out.print("Library in which the file exists: ");
        library = inputStream.readLine();

        System.out.print("File name: ");
        file = inputStream.readLine();

        System.out.print("Member name (press enter for first member): ");
        member = inputStream.readLine();
        if (member.equals(""))
        {
          member = "*FIRST";
        }

        System.out.println();
      }
      catch (Exception e)
      {
        System.out.println("Error obtaining user input.");
        e.printStackTrace();
        System.exit(0);
      }

      // Create AS400 object and connect for the record level access service.
      AS400 system = new AS400(systemName);
      try
      {
        system.connectService(AS400.RECORDACCESS);
      }
      catch (Exception e)
      {
        System.out.println("Unable to connect for record level access.");
        System.out.println("Check the readme file for special instructions regarding record
                            level access");
        e.printStackTrace();
        System.exit(0);
      }

      // Create a QSYSObjectPathName object to obtain the integrated file system path name form
      // of the file to be displayed.
      QSYSObjectPathName filePathName = new QSYSObjectPathName(library, file, member, "MBR");

      // Create a SequentialFile object representing the file to be displayed
      SequentialFile theFile = new SequentialFile(system, filePathName.getPath());

      // Retrieve the record format for the file
      AS400FileRecordDescription recordDescription =
        new AS400FileRecordDescription(system, filePathName.getPath());
      try
      {
        RecordFormat[] format = recordDescription.retrieveRecordFormat();

        // Set the record format for the file
        theFile.setRecordFormat(format[0]);

        // Open the file for reading.Read 100 records at a time if possible.
theFile.open(AS400File.READ_ONLY, 100, AS400File.COMMIT_LOCK_LEVEL_NONE);

        // Display each record in the file
        System.out.println("Displaying file " + library.toUpperCase() + "/"
                           + file.toUpperCase() + "(" + theFile.getMemberName().trim() + "):");

        Record record = theFile.readNext();
```

```
        while (record != null)
        {
          System.out.println(record);
          record = theFile.readNext();
        }
        System.out.println();

        // Close the file
        theFile.close();

        // Disconnect from the record level access service
        system.disconnectService(AS400.RECORDACCESS);
      }
      catch (Exception e)
      {
        System.out.println("Error occurred attempting to display the file.");
        e.printStackTrace();

        try
        {
          // Close the file
          theFile.close();
        }
        catch(Exception x)
        {
        }

        // Disconnect from the record level access service
        system.disconnectService(AS400.RECORDACCESS);
        System.exit(0);
      }

      // Make sure that the application ends; see readme for details
      System.exit(0);
    }
  }
```

範例：使用記錄層次存取類別來類別檔案中的記錄

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////
//
// Record-Level Access example.This program uses the record-level
// access classes to read records from a file on the server.
//
// Command syntax:
//java RLReadFile system
//
// This program reads the records from CA/400's sample database file
// (QCUSTCDT in library QIWS).If you change this example to update
// records, make a copy of QCUSTCDT and update the copy.
//
// This source is an example of IBM Toolbox for Java "Record-level access".
//
///////////////////////////////////////////////////////////////////////////


import java.io.*;
 import java.util.*;
import java.math.*;
import com.ibm.as400.access.*;

public class RLReadFile extends Object
{
   public static void main(String[] parameters)
   {
```

```
      String system = "";

      // Continue only if a system name was specified.

      if (parameters.length >= 1)
      {

         try
         {

            // Assume the first parameter is the system name.

            system = parameters[0];

            // Create an AS400 object for the server that has the file.

            AS400 as400 = new AS400(system);


            // Create a record description for the file.The file is QCUSTCDT
// in library QIWS.

            ZonedDecimalFieldDescription customerNumber =
                         new ZonedDecimalFieldDescription(new AS400ZonedDecimal(6,0),
                                                          "CUSNUM");
            CharacterFieldDescription lastName =
                         new CharacterFieldDescription(new AS400Text(8, as400), "LSTNAM");

            CharacterFieldDescription initials =
                         new CharacterFieldDescription(new AS400Text(3, as400), "INIT");

            CharacterFieldDescription street =
                         new CharacterFieldDescription(new AS400Text(13, as400), "STREET");

            CharacterFieldDescription city =
                         new CharacterFieldDescription(new AS400Text(6, as400), "CITY");

            CharacterFieldDescription state =
                         new CharacterFieldDescription(new AS400Text(2, as400), "STATE");

            ZonedDecimalFieldDescription zipCode =
                         new ZonedDecimalFieldDescription(new AS400ZonedDecimal(5,0),
                                                          "ZIPCOD");
            ZonedDecimalFieldDescription creditLimit =
                         new ZonedDecimalFieldDescription(new AS400ZonedDecimal(4,0),
                                                          "CDTLMT");
            ZonedDecimalFieldDescription chargeCode =
                         new ZonedDecimalFieldDescription(new AS400ZonedDecimal(1,0),
                                                          "CHGCOD");
            ZonedDecimalFieldDescription balanceDue =
                         new ZonedDecimalFieldDescription(new AS400ZonedDecimal(6,2),
                                                          "BALDUE");
            ZonedDecimalFieldDescription creditDue =
                         new ZonedDecimalFieldDescription(new AS400ZonedDecimal(6,2),
                                                          "CDTDUE");


            // The record format name must be specified for a DDM file.
            // In the case of the QCUSTCDT file, its record format is called CUSREC.

            RecordFormat qcustcdt = new RecordFormat("CUSREC");

            qcustcdt.addFieldDescription(customerNumber);
            qcustcdt.addFieldDescription(lastName);
            qcustcdt.addFieldDescription(initials);
            qcustcdt.addFieldDescription(street);
```

```
            qcustcdt.addFieldDescription(city);
            qcustcdt.addFieldDescription(state);
            qcustcdt.addFieldDescription(zipCode);
            qcustcdt.addFieldDescription(creditLimit);
            qcustcdt.addFieldDescription(chargeCode);
            qcustcdt.addFieldDescription(balanceDue);
            qcustcdt.addFieldDescription(creditDue);


            // Create the sequential file object that represents the
            // file on the server.We use a QSYSObjectPathName object
// to get the name of the file into the correct format.

            QSYSObjectPathName fileName = new QSYSObjectPathName("QIWS",
                                                                 "QCUSTCDT",
                                                                 "FILE");

            SequentialFile file = new SequentialFile(as400, fileName.getPath());


            // Let the file object know the format of the records.

            file.setRecordFormat(qcustcdt);


            // Open the file for read-only access.Specify a blocking
// factor of 10 (the file object will get 10 records when
            // it accesses the server for data).Do not use commitment
// control.

            file.open(SequentialFile.READ_ONLY,
                      10,
                      SequentialFile.COMMIT_LOCK_LEVEL_NONE);


            // Read the first record of the file.

            Record data = file.readNext();


            // Loop while there are records in the file (while we have not
            // reached end-of-file).

             while (data != null)
            {

                // Display the record only if balance due is greater than
                // zero.In that case display the customer name and
 // the balance due.The following code pulls fields out
 // of the record by field name.As the field is retrieved
 // from the record it is converted from server format to
               // Java format.

               if (((BigDecimal)data.getField("BALDUE")).floatValue() > 0.0)
               {
                  System.out.print((String) data.getField("INIT") + "  ");
                  System.out.print((String) data.getField("LSTNAM") + "  ");
                  System.out.println((BigDecimal) data.getField("BALDUE"));
               }

               // Read the next record in the file.

               data = file.readNext();
            }


            // When there are no more records to read, disconnect from the server.
```

```
            as400.disconnectAllServices();
         }

         catch (Exception e)
         {

            // If any of the above operations failed, print an error message
            // and output the exception.

            System.out.println("Could not read the file");
            System.out.println(e);
         }
      }


      // Display help text when parameters are incorrect.

      else
      {
         System.out.println("");
         System.out.println("");
         System.out.println("");
         System.out.println("Parameters are not correct.  Command syntax is:");
         System.out.println("");
         System.out.println("   RLReadFile as400");
         System.out.println("");
         System.out.println("Where");
         System.out.println("");
         System.out.println("   as400 = system that contains the file");
         System.out.println("");
         System.out.println("For example:");
         System.out.println("");
         System.out.println("   RLReadFile mySystem");
         System.out.println("");
         System.out.println("");
         System.out.println("Note, this program reads data base file QIWS/QCUSTCDT.  ");
         System.out.println("");
         System.out.println("");
      }

      System.exit(0);

   }
}
```

範例：使用記錄層次存取類別按金鑰讀取記錄

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// Record-Level Access example.This program uses the record-level
// access classes to read records by key from a file on the server.
// The user will be prompted for the server name to which to run and
// the library in which to create file QCUSTCDTKY.
//
// Command syntax:
//java RLKeyedFileExample
//
// This program will copy the records from the iSeries Access for Windows sample
// database file (QCUSTCDT in library QIWS) to file QCUSTCDTKY which has
// the same format as QIWS/QCUSTCDT but has set the CUSNUM field as the key
// for the file.
//
// This source is an example of IBM Toolbox for Java "Record-level access".
//
```

```
////////////////////////////////////////////////////////////////////////////


import java.io.*;
 import java.util.*;
import java.math.*;
import com.ibm.as400.access.*;

public class RLKeyedFileExample
{
  public static void main(String[] parameters)
  {

    // Created a reader to get input from the user
    BufferedReader inputStream = new BufferedReader(new InputStreamReader(System.in),1);

    // Declare variables to hold the system name, library, file and member names
    String systemName = "";
    String library = "";

    // Get the system name from the user
    System.out.println();
     try
    {
      System.out.print("System name: ");
      systemName = inputStream.readLine();

      System.out.print("Library in which to create file QCUSTCDTKY: ");
      library = inputStream.readLine();
    }
    catch (Exception e)
    {
      System.out.println("Error obtaining user input.");
      e.printStackTrace();
      System.exit(0);
    }

    // Create AS400 object and connect for the record level access service.
    AS400 system = new AS400(systemName);
    try
    {
      system.connectService(AS400.RECORDACCESS);
    }
    catch (Exception e)
    {
      System.out.println("Unable to connect for record level access.");
      System.out.println("Check the readme file for special instructions regarding record
                          level access");
      e.printStackTrace();
      System.exit(0);
    }

    RecordFormat qcustcdtFormat = null;
    try
    {
      // Create the RecordFormat object for creating the file.The record format for the new
// file will be the same as the record format for file QIWS/QCUSTCDT.However we will
// make the CUSNUM field a key field.
      AS400FileRecordDescription recordDescription =
        new AS400FileRecordDescription(system, "/QSYS.LIB/QIWS.LIB/QCUSTCDT.FILE");

      // There is only one record format for the file, so take the first (and only) element
      // of the RecordFormat array returned as the RecordFormat for the file.
      System.out.println("Retrieving record format of QIWS/QCUSTCDT...");
      qcustcdtFormat = recordDescription.retrieveRecordFormat()[0];
      // Indicate that CUSNUM is a key field
      qcustcdtFormat.addKeyFieldDescription("CUSNUM");
```

```
    }
    catch (Exception e)
    {
      System.out.println("Unable to retrieve record format from QIWS/QCUSTCDT");
      e.printStackTrace();
      System.exit(0);
    }

    // Create the keyed file object that represents the
    // file we will create on the server.We use a QSYSObectPathName object
// to get the name of the file into the correct format.
    QSYSObjectPathName fileName = new QSYSObjectPathName(library,
                                                         "QCUSTCDTKY",
                                                         "*FILE",
                                                         "MBR");
    KeyedFile file = new KeyedFile(system, fileName.getPath());

    try
    {
      System.out.println("Creating file " + library + "/QCUSTCDTKY...");
      // Create the file using the qcustcdtFormat object
      file.create(qcustcdtFormat, "Keyed QCUSTCDT file");

      // Populate the file with the records contained in QIWS/QCUSTCDT
      copyRecords(system, library);

      // Open the file for read-only access.Because we will be randomly
// accessing the file, specify a blocking factor of 1.The
// commit lock level parameter will be ignored since commitment
      // control has not been started.
      file.open(AS400File.READ_ONLY,
                1,
                AS400File.COMMIT_LOCK_LEVEL_NONE);

      // Assume that we want to display the information for customers
      // 192837, 392859 and 938472
      // The CUSNUM field is a zoned decimal field of length 6 with
      // no decimal positions.Therefore, the key field value is
// represented with a BigDecimal.
      BigDecimal[] keyValues = {new BigDecimal(192837), new BigDecimal(392859), new BigDecimal(938472)};
// Create the key for reading the records.The key for a KeyedFile
// is specified with an Object[]
      Object[] key = new Object[1];

      Record data = null;
      for (int i = 0; i < keyValues.length; i++)
      {
        // Setup the key for reading
        key[0] = keyValues[i];

        // Read the record for customer number keyValues[i]
        data = file.read(key);
        if (data != null)
        {
          // Display the record only if balance due is greater than
          // zero.  In that case display the customer name and
          // the balance due.The following code pulls fields out
// of the record by field name.As the field is retrieved
// from the record it is converted from the server format to
          // Java format.
          if (((BigDecimal)data.getField("BALDUE")).floatValue() > 0.0)
          {
            System.out.print((String) data.getField("INIT") + "  ");
            System.out.print((String) data.getField("LSTNAM") + "  ");
            System.out.println((BigDecimal) data.getField("BALDUE"));
          }
        }
```

```
      }

      // All done with the file
      file.close();

      // Get rid of the file from the user's system
      file.delete();
    }
    catch (Exception e)
    {
      System.out.println("Unable to create/read from QTEMP/QCUSTCDT");
      e.printStackTrace();
      try
      {
        file.close();
        // Get rid of the file from the user's system
        file.delete();
      }
      catch(Exception x)
      {
      }
    }

    // All done with record level access; disconnect from the
    // record-level access server.
    system.disconnectService(AS400.RECORDACCESS);
    System.exit(0);
  }

  public static void copyRecords(AS400 system, String library)
  {
    // Use the CommandCall class to run the CPYF command to copy the records
    // in QIWS/QCUSTCDT to QTEMP/QCUSTCDT
    CommandCall c = new CommandCall(system, "CPYF FROMFILE(QIWS/QCUSTCDT) TOFILE("
                                    + library + "/QCUSTCDTKY) MBROPT(*REPLACE)");
    try
    {
      System.out.println("Copying records from QIWS/QCUSTCDT to "
                         + library + "/QCUSTCDTKY...");
      c.run();
      AS400Message[] msgs = c.getMessageList();
      if (!msgs[0].getID().equals("CPC2955"))
      {
        System.out.println("Unable to populate " + library + "/QCUSTCDTKY");
        for (int i = 0; i < msgs.length; i++)
        {
          System.out.println(msgs[i]);
        }
        System.exit(0);
      }
    }
    catch (Exception e)
    {
      System.out.println("Unable to populate " + library + "/QCUSTCDTKY");
      System.exit(0);
    }
  }
}
```

範例：使用 UserList 列出給定的群組中的所有使用者

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// User list example.This program lists all of the users in a given
// group.
```

```
//
// Command syntax:
//UserListExample system group
//
// This source is an example of IBM Toolbox for Java "UserList".
//
//////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import java.util.Enumeration;

public class UserListExample
{


     public static void main(String[] args)
    {
        // If a system and group were not specified, then display
        // help text and exit.
        if (args.length != 2)
        {
            System.out.println("Usage:  UserListExample system group");
            return;
        }

        try
        {
            // Create an AS400 object.  The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);

            // The group name was passed as the second command line
            // argument.
            String groupName = args[1];

            // Create the user list object.
            UserList userList = new UserList (system);

            // Get a list of the users in the given group.
            userList.setUserInfo (UserList.MEMBER);
            userList.setGroupInfo (groupName);
            Enumeration enum = userList.getUsers ();

            // Iterate through the list and print out the
            // users' names and descriptions.
            while (enum.hasMoreElements ())
            {
                User u = (User) enum.nextElement ();
                System.out.println ("User name:   " + u.getName ());
                System.out.println ("Description: " + u.getDescription ());
                System.out.println ("");
            }

        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
        }

        System.exit (0);
    }


}
```

範例：JavaBeans

本節將列出 IBM Toolbox for Java Bean 資訊所提供的所有程式碼範例。

- 範例：當您連線到系統及切斷連線，並執行指令時，使用接收器來列印備註
- 範例：使用 Applet 及 IBM VisualAge for Java 來建立執行指令的按鈕

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：IBM Toolbox for Java Bean 程式碼

下面範例建立 AS400 物件和 CommandCall 物件，然後在物件中登記接收器。當伺服器連線或斷線以及當 CommandCall 物件完成指令的執行時，物件上的接收程式會列印註解。

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
///////////////////////////////////////////////////////////////////////////////
//
// Beans example.  This program uses the JavaBeans support in the
// IBM Toolbox for Java classes.
//
// Command syntax:
//    BeanExample
//
///////////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.CommandCall;
import com.ibm.as400.access.ConnectionListener;
import com.ibm.as400.access.ConnectionEvent;
import com.ibm.as400.access.ActionCompletedListener;
import com.ibm.as400.access.ActionCompletedEvent;

class BeanExample
{
        AS400       as400_  = new AS400();
        CommandCall cmd_    = new CommandCall( as400_ );

        BeanExample()
    {
                // Whenever the system is connected or disconnected print a
                // comment. Do this by adding a listener to the AS400 object.
        // When a system is connected or disconnected, the AS400 object
                // will call this code.

                as400_.addConnectionListener
                (new ConnectionListener()
          {
             public void connected(ConnectionEvent event)
             {
```

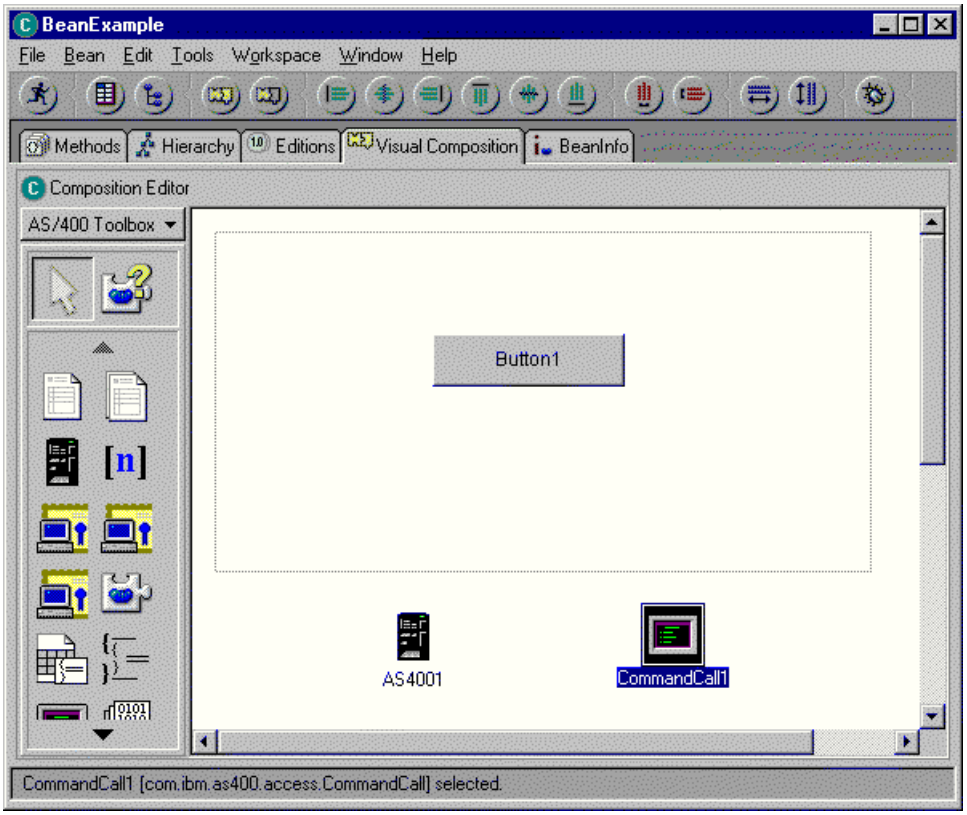
```
                System.out.println( "System connected." );
            }
            public void disconnected(ConnectionEvent event)
            {
                System.out.println( "System disconnected." );
            }
          }
        );

        // Whenever a command runs to completion print a comment. Do this
        // by adding a listener to the commandCall object. The commandCall
        // object will call this code when it runs a command.

        cmd_.addActionCompletedListener(
            new ActionCompletedListener()
            {
                public void actionCompleted(ActionCompletedEvent event)
                {
                    System.out.println( "Command completed." );
                }
            }
        );
    }

    void runCommand()
    {
        try
        {
            // Run a command. The listeners will print comments when the
            // system is connected and when the command has run to
            // completion.
            cmd_.run( "TESTCMD PARMS" );
        }
        catch (Exception ex)
        {
            System.out.println( ex );
        }
    }

    public static void main(String[] parameters)
    {
       BeanExample be = new BeanExample();

       be.runCommand();

       System.exit(0);
    }
}
```

範例：使用視覺化 bean 建置器來建立 bean

此範例使用「IBM VisualAge for Java 企業版 V2.0 組合編輯程式」，但是其他視覺化 Bean 建置器也一樣。本範例將建立按鈕的 Applet，當按下按鈕後，即會在 iSeries 伺服器上執行指令。

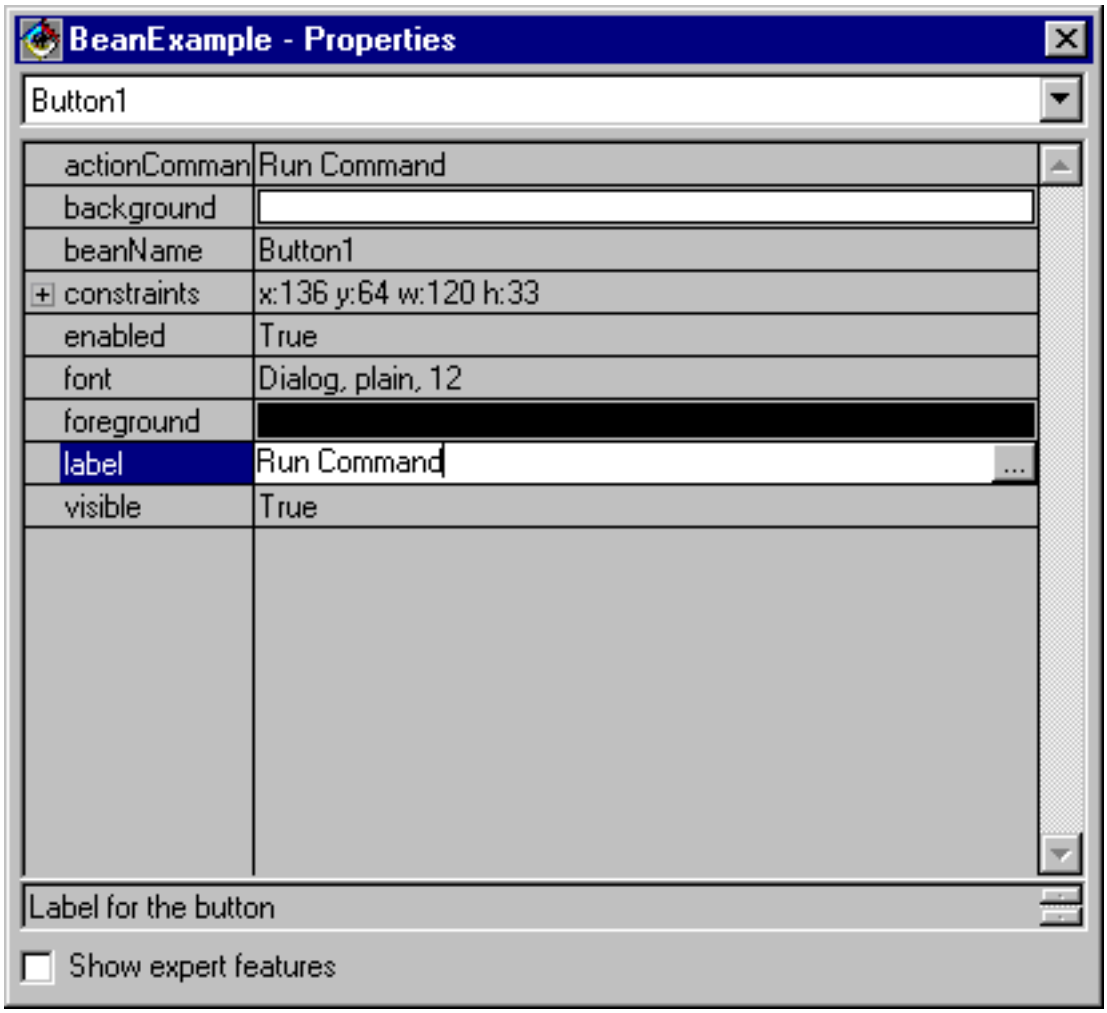
- 在 Applet 上拖放「按鈕」。(在圖形 1 中的 Visual Composition 標籤左邊的 bean 建立器中可找到此「按鈕」。)
- 將 CommandCall bean 與 AS400 bean 拖到小型程式的外面。(在圖形 1 中的 Visual Composition 標籤左邊的 bean 建立器中可找到 bean。)

圖 1：VisualAge Visual Composition Editor 視窗 - gui.BeanExample



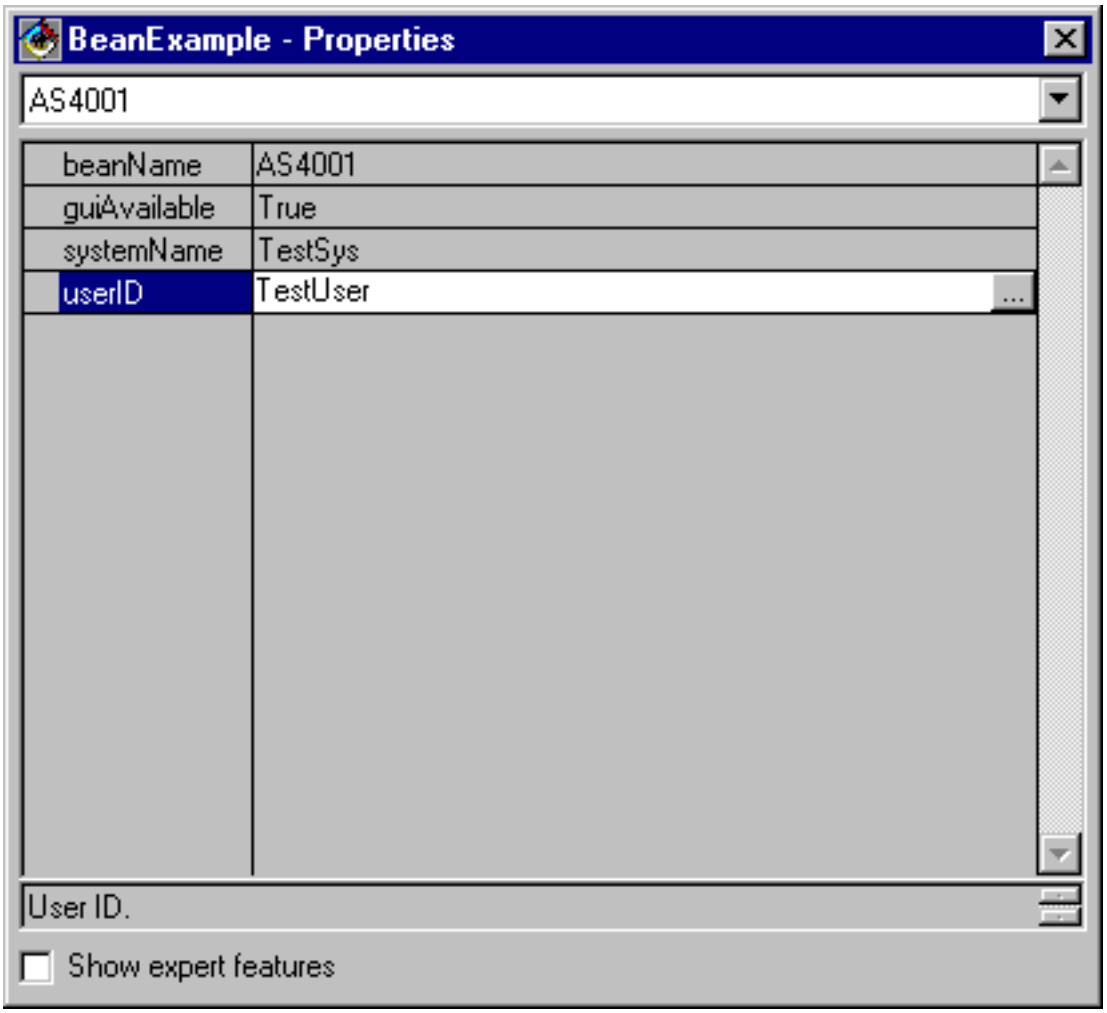
- 編輯 bean 特性。(若要編輯，請選取此 bean，然後以滑鼠右鍵按一下來顯示視窗，您將在視窗中「內容」這個選項。)
  - 將 Button 標籤變更爲 **Run command**，如圖 2 所示。

    圖 2：將按鈕標籤變更為 Run command 指令



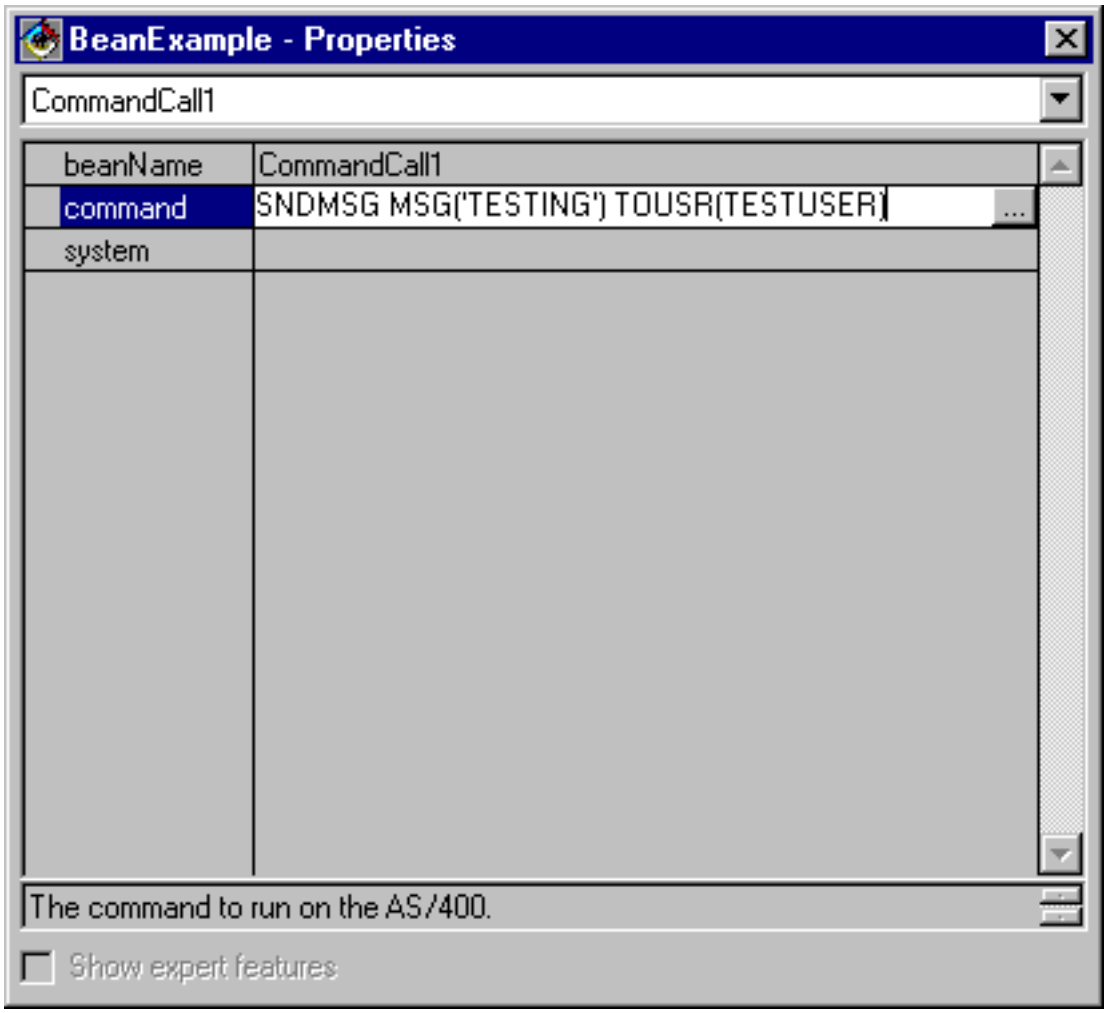
– 將 AS400 bean 的系統名稱變更爲 **TestSys**
– 將 AS400 bean 的 user ID (使用者 ID) 變更爲 **TestUser**，如圖形 3 所示。

Figure 3: Changing the name of the user ID to TestUser



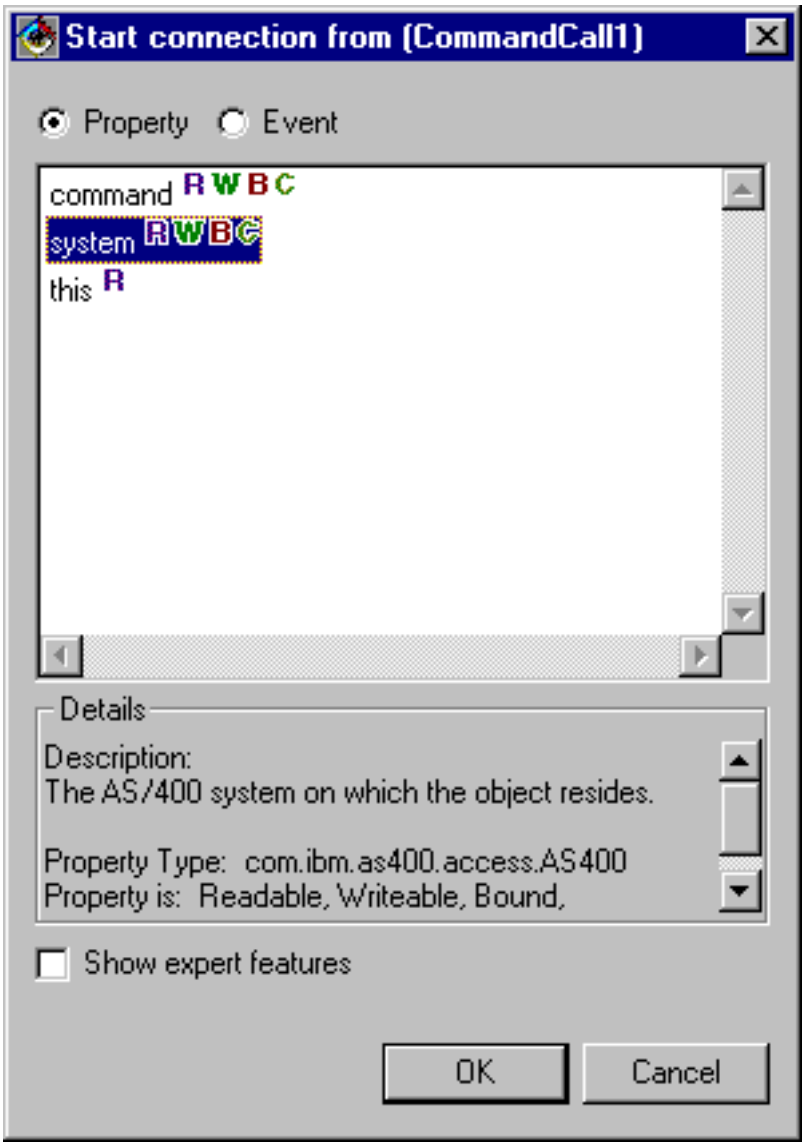
– 將 CommandCall Bean 的指令變更爲 **SNDMSG MSG('Testing') TOUSR('TESTUSER')**，如圖 4 所示。

圖 4：變更 CommandCall Bean 的指令



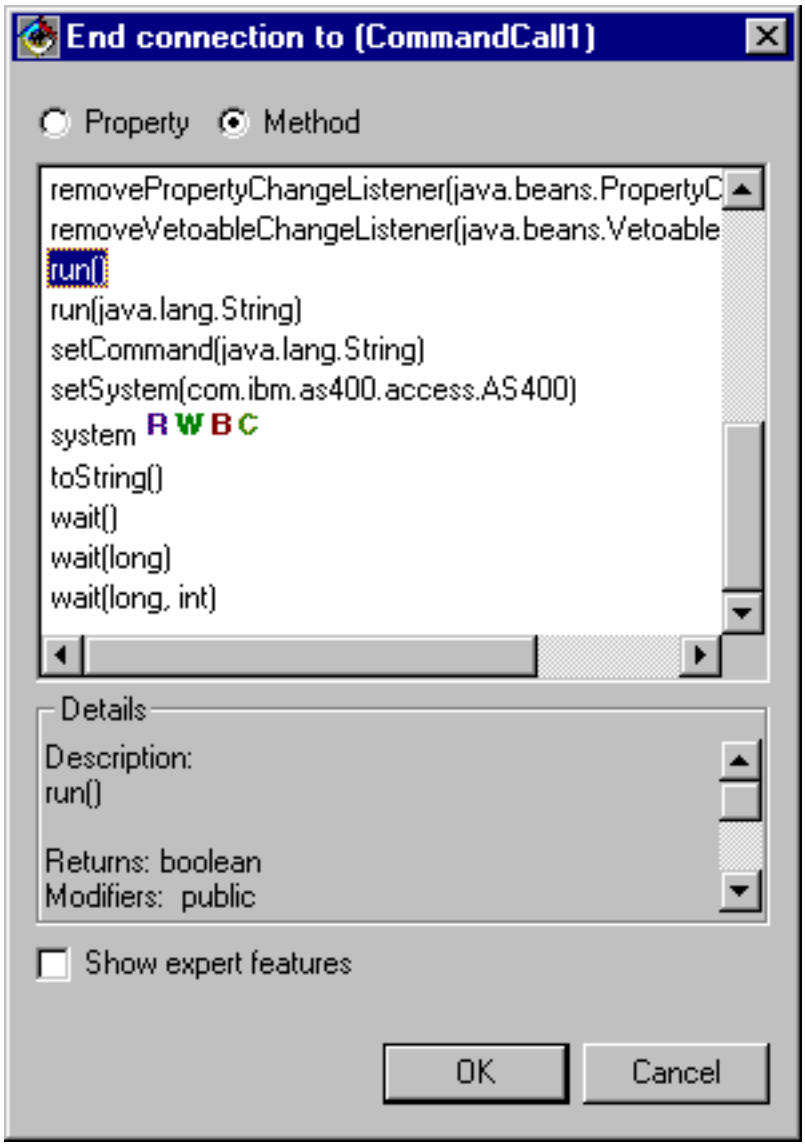
- 連接 AS400 bean 到 CommandCall bean。您用來執行此動作的方法會 因 bean 建立器而不同。在此範例中，請執行下列步驟：
  - 選取 CommandCall bean，然後按一下右滑鼠按鈕
  - 選取 **Connect**
  - 選取 **Connectable Features**
  - 從圖形 5 中顯示的特性列性中選取 **system**。
  - 選取 AS400 bean
  - 從顯示於 AS400 Bean 上的蹦現功能表選取 **this**。

圖 5：連接 AS400 Bean 到 CommandCall Bean



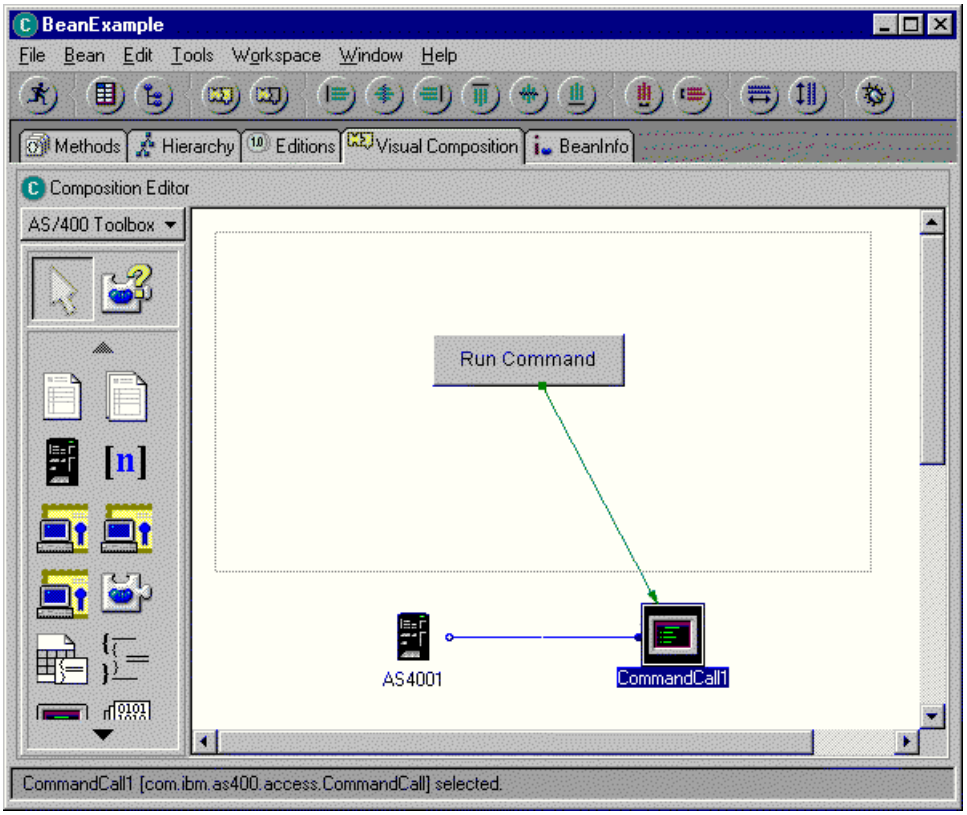
- 使按鈕連線到 CommandCall bean。
  - 選取 Button bean，然後按一下右滑鼠按鈕
  - 選取 **Connect**
  - 選取 **actionPerformed**
  - 選取 CommandCall bean
  - 從出現的蹦現功能表選取 **Connectable Features**
  - 從圖形 6 顯示的方法清單中選取 **run()**。

圖 6：連線方法與按鈕



完成後，VisualAge Visual Composition Editor 視窗可能如「圖 7」所示。

圖 7：VisualAge Visual Composition Editor 視窗- 完成的 Bean 範例



範例：Commtrace 類別

此網頁鏈結至 IBM Toolbox for Java Commtrace 類別之文件所提供的程式碼範例。

- 第 173 頁的『範例：使用 commtrace 類別』

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

Graphical Toolbox 範例

這些範例將說明如何實作 Graphical Toolbox 中的工具，以完成您自己的 UI 程式。

- 建構與顯示畫面：建構一個簡式畫面，以此例來對 Graphical Toolbox 環境的基本特性及作業做一個整體的說明
- 建立及顯示畫面：當畫面與內容檔位於相同的目錄中時建立並顯示畫面
- 建構全功能對話框：建構全功能的對話框 (當您施行了能爲畫面提供資料的 DataBeans， 並識別 PDML 中的屬性之後)

- 使用動態畫面管理程式調整畫面大小：動態畫面管理程式於執行時間動態調整畫面大小的方式
- 可編輯的組合框：爲可編輯的組合框撰寫資料 Bean 程式碼

下面範例會說明 GUI Builder 如何協助您建立各種 GUI 元素：
- 畫面：建立畫面範例，以及執行畫面的資料 Bean 程式碼
- 重疊窗格：建立重疊窗格，及其最終的外觀
- 內容表：建立內容表，及其最終的外觀
- 分割窗格：建立分割窗格，及其最終的外觀
- 標籤窗格：建立標籤窗格，及其最終的外觀
- 精靈：建立精靈，以及最終產品的外觀
- 工具列：建立工具列，及其最終的外觀
- 功能表列：建立功能表列，及其最終的外觀
- 說明：建立「說明文件」，並將「說明文件」分成不同的主題頁。同時，請參閱編輯由 GUI Builder 產生的說明文件
- 範例：說明整個 PDML 程式看起來的樣子，包括畫面、內容表、精靈、選取/取消選取及功能表選項。

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

範例：用 GUI Builder 來建構畫面

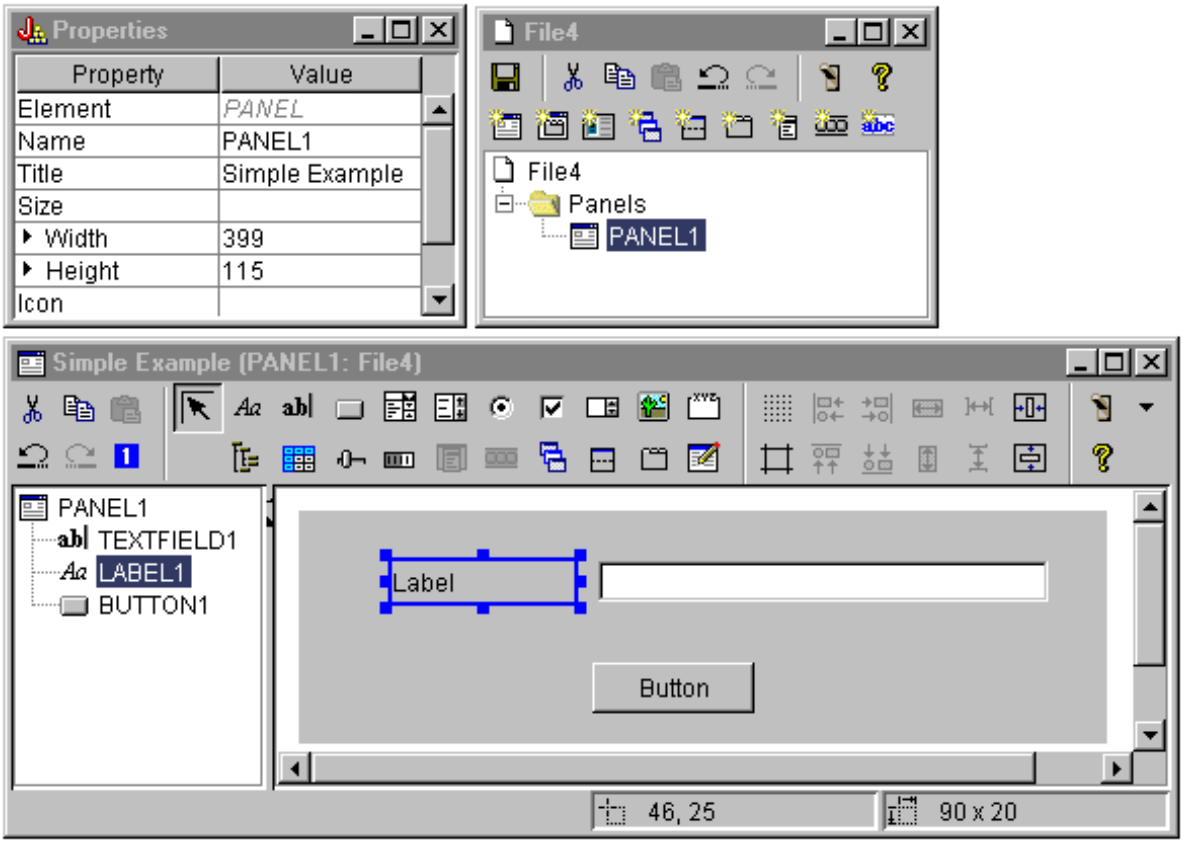
註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

此範例會藉著建構一個簡單的畫面來示範如何使用 Graphical Toolbox。 這是一個概觀，說明 Graphical Toolbox 環境的基本特性及作業。在顯示如何建構畫面後，範例會繼續執行，並說明如何建置一個顯示畫面的小型 Java 應用程式。在此範例中，使用者會在文字欄位中輸入資料，並按一下 **Close** 按鈕。然後應用程式會將資料傳給 Java 主控台。

建構畫面

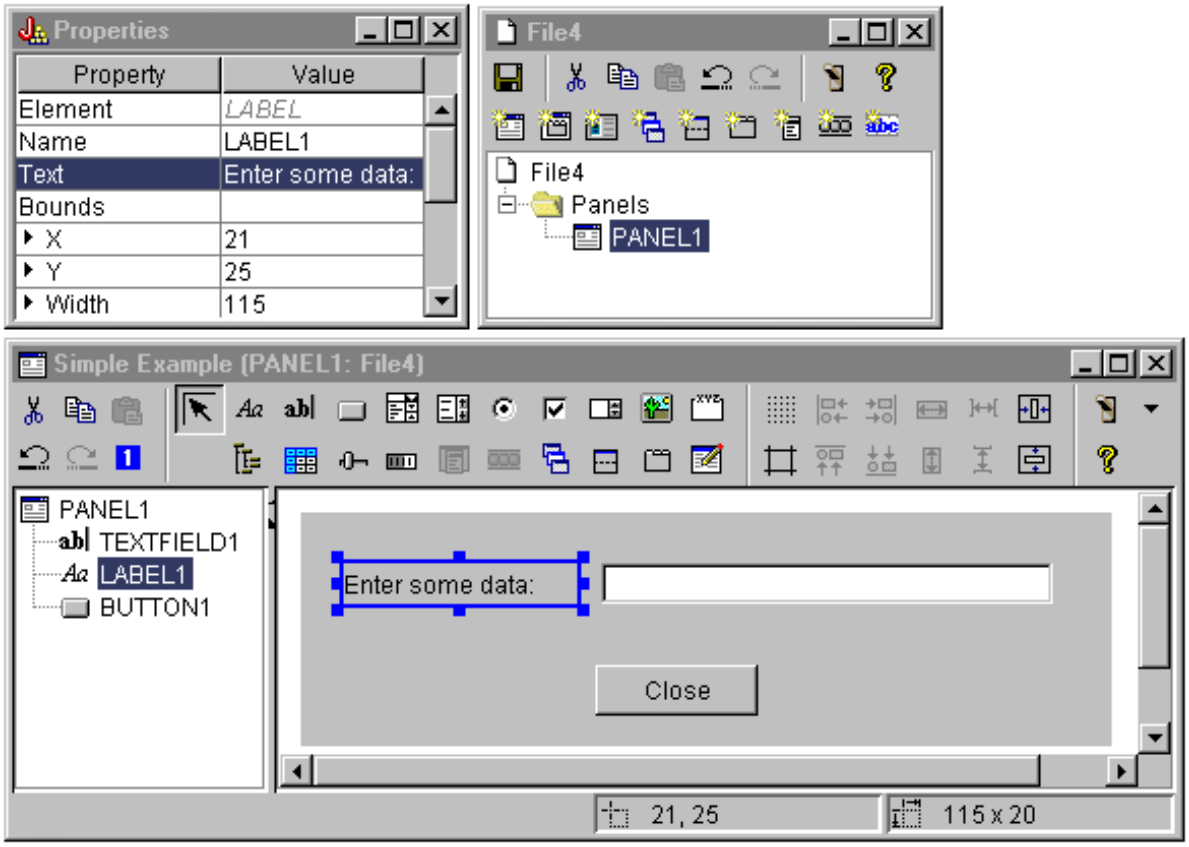
當您啓動 GUI Builder 時，會出現 Properties 及 GUI Builder 視窗。建立一個名爲 MyGUI.pdml 的新檔案。就此例而言，會插入一個新的畫面。 在 File Builder 視窗中，按一下 Insert Panel 圖示。 它的名稱爲 ″PANEL1″。在 Properties 視窗中修改資訊，以變更標題；在 Titile 欄位中鍵入 Simple Example。以滑鼠選取三個預設的按鈕，然後按一下 Delete 鍵即可除去三個預設按鈕。 使用 Panel Builder 視窗中的按鈕，新增圖 1 所示的三個元素：一個標籤、一個文字欄位及一個按鈕。

圖 1：GUI Builder 視窗：開始建構畫面



選取標籤後， 您可以在 Properties 視窗中變更它的文字。本範例中，也對按鈕作出變更，文字改爲 Close。

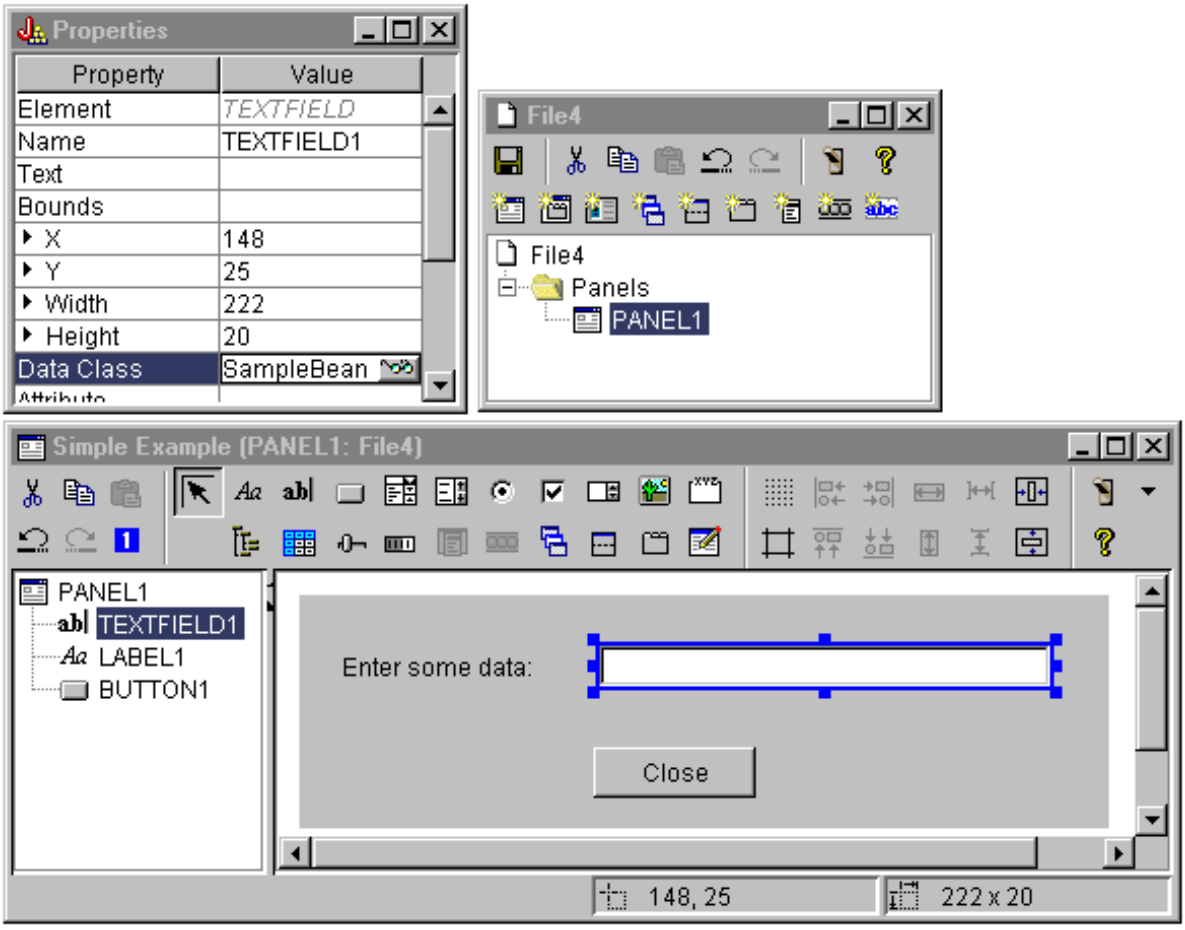
圖 2：GUI Builder 視窗：變更 Porperties 視窗中的文字



Text 欄位

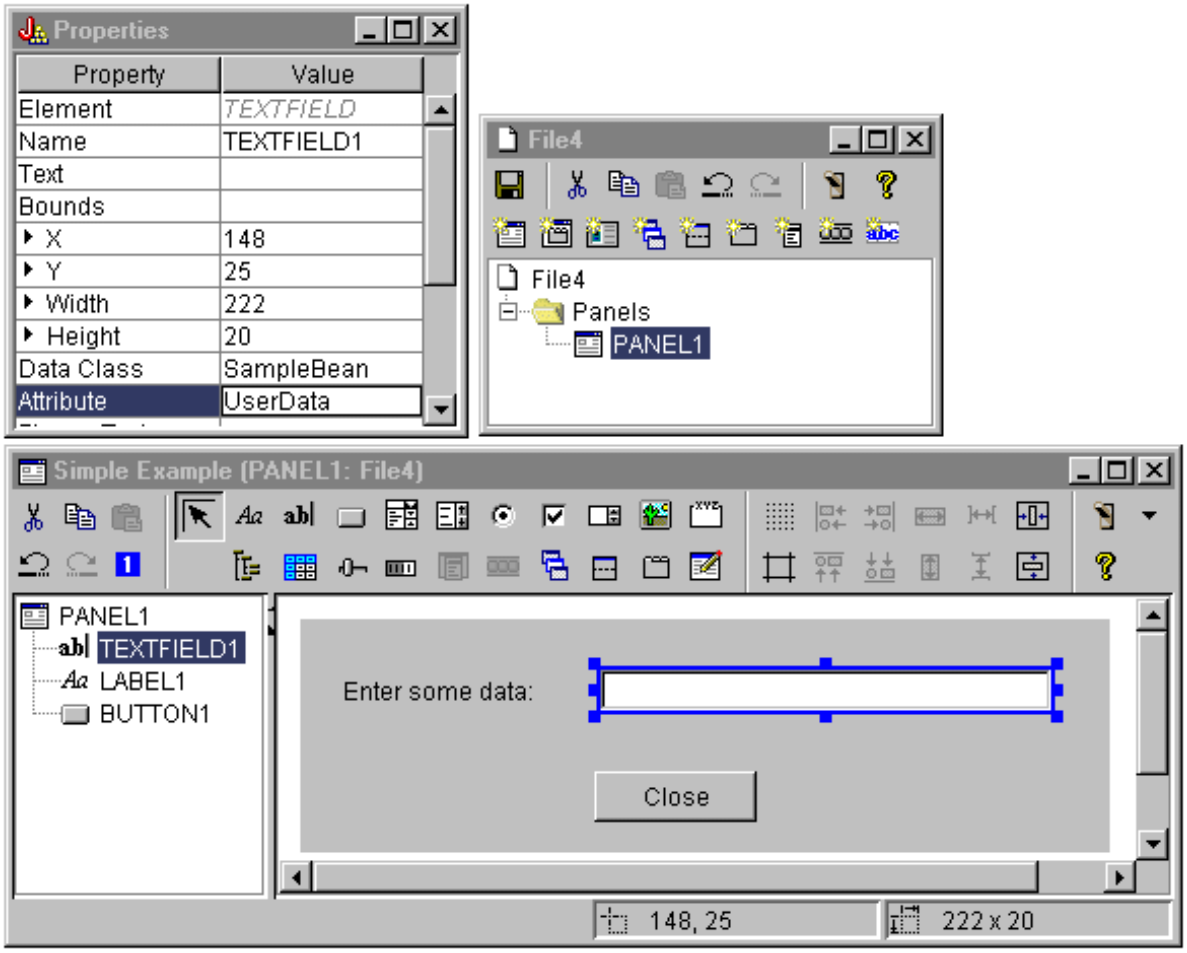
Text 欄位會包含資料，因此，您可以設定數種特性，讓 GUI Builder 執行 一些其它的工作。就本例而言，您可以將 Data Class 內容設定爲 bean 類別 的名稱，命名爲 **SampleBean**。此 databean 會提供此文字欄位的資料。

圖 3：GUI Builder 視窗：設定 Data Class 內容



將 Attribute 特性設定爲 bean 特性的名稱 (其中含有資料)。 在此情況下，名稱爲 **UserData**。

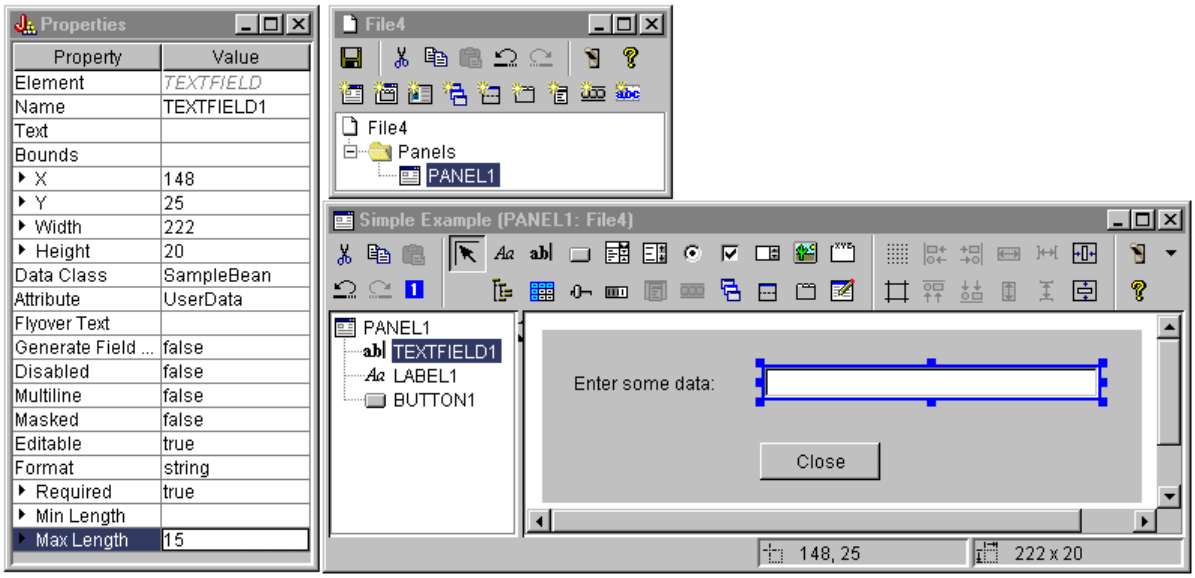
圖 4：GUI Builder 視窗：設定 Attribute 內容



請遵循上述步驟，將 **UserData** 連結到此 Text 欄位。在執行時間，Graphical Toolbox 會呼叫 **SampleBean.getUserData**，取得此欄位的起始值。然後，當畫面關閉時，會呼叫 **SampleBean.setUserData**，將修改過的傳回應用程式。

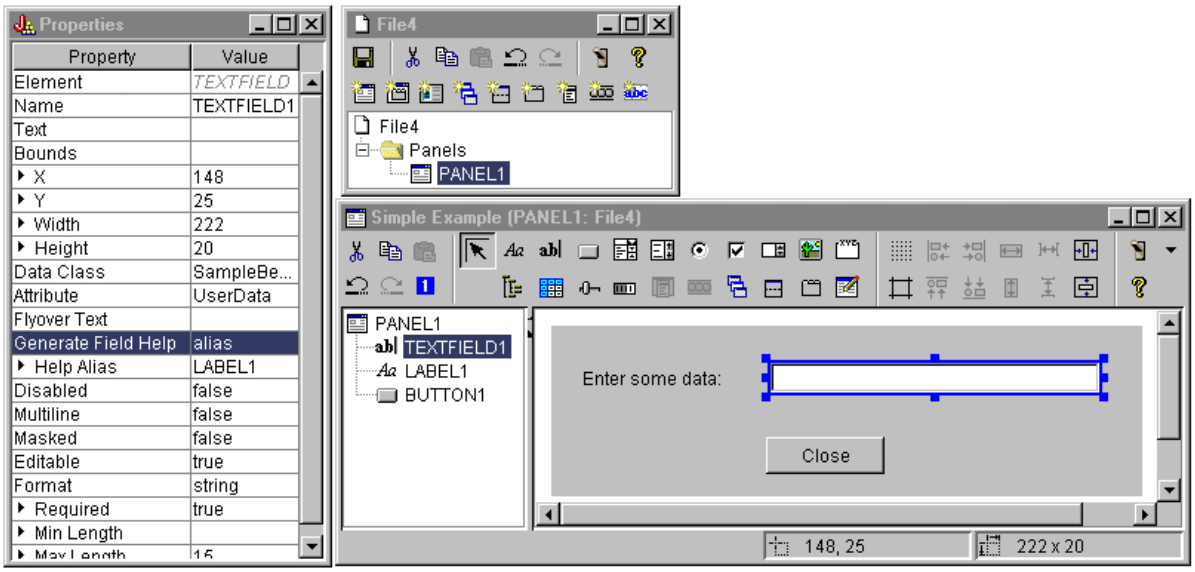
指定使用者必須提供某些資料，且該資料必須是一個字串，字串的長度上限爲 15 個字元。

圖 5：GUI Builder 視窗：設定文字欄位的最大長度



表示文字欄位的上下文相關的說明是與標籤 "Enter some data" 相關的說明主題。

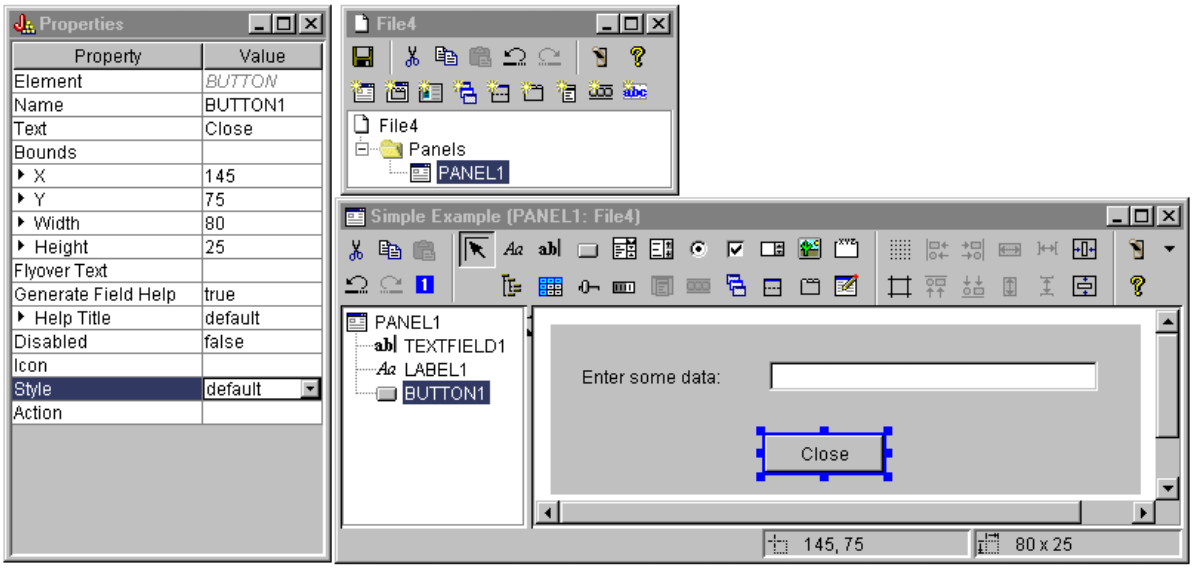
圖 6：GUI Builder 視窗：設定文字欄位的環境定義相關說明



按鈕

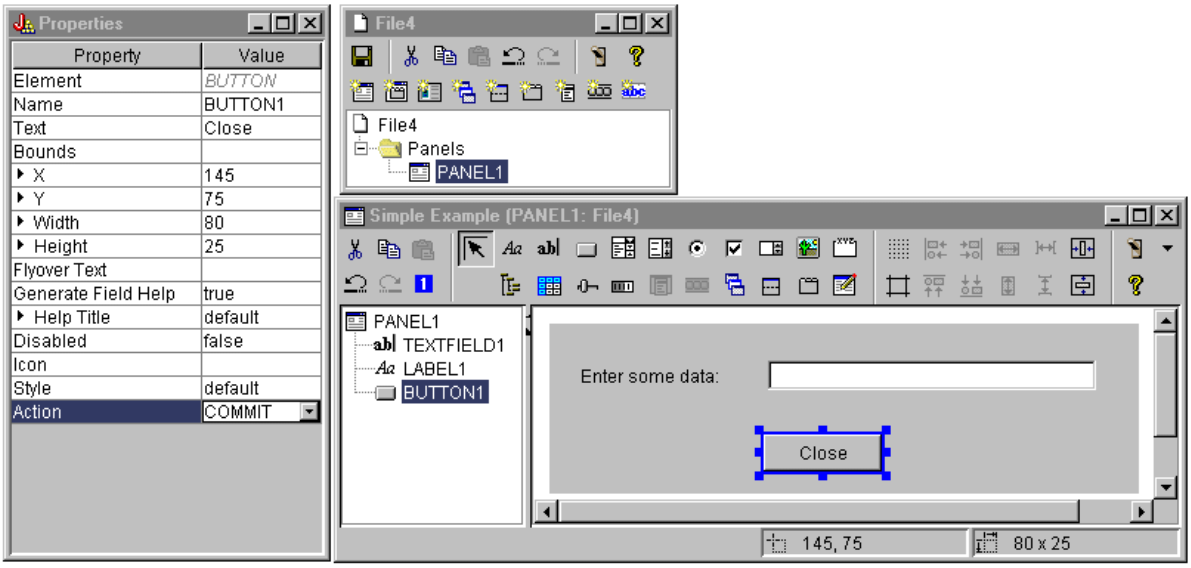
修改此樣式特性，以強調按鈕的預設值。

圖 7：GUI Builder 視窗：設定 Style 內容以強調按鈕的預設值



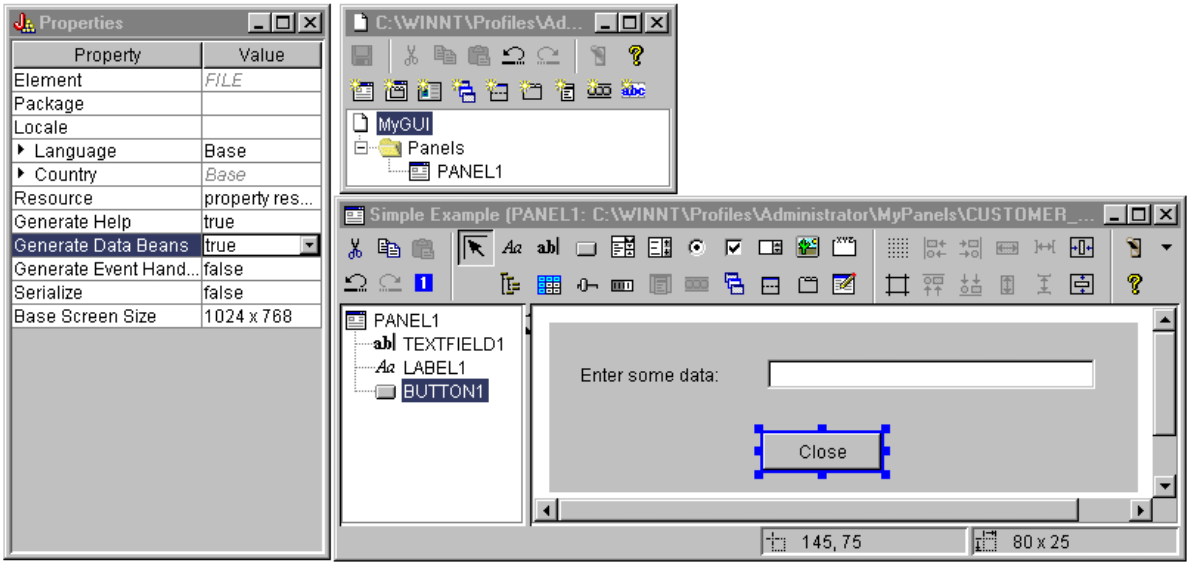
將 ACTION 特性設定爲 COMMIT，這會造成在選取按鈕時，會呼叫 bean 中的 `setUserData`方法。

圖 8：GUI Builder 視窗：將 Action 內容設定為 COMMIT



在您儲存畫面之前，請設定 PDML 檔案層次的內容，以產生線上說明架構及 Java Bean。然後您可以在 GUI Builder 視窗中按一下  圖示，儲存檔案。在系統提示時，請指定 **MyGUI.pdml** 的檔名。

圖 9：GUI Builder 視窗：設定內容以產生線上說明架構及 Java Bean



產生檔案

在您儲存畫面定義後，您可以察看由 GUI Builder 所產生的檔案。**PDML 檔** 這是 **MyGUI.pdml** 的內容，可讓您瞭解 Panel Definition Markup Language 的運作方式。 因爲您只能經由 Graphical Toolbox 所提供的工具使用 PDML，所以不需要詳細地瞭解此檔案的格式：

```
<!-- Generated by GUI Builder -->
<PDML version="2.0" source="JAVA" basescreensize="1280x1024">

 <PANEL name="PANEL1">
  <TITLE>PANEL1<TITLE>
  <SIZE>351,162<</SIZE>
  <LABEL name="LABEL1"">
   <TITLE>PANEL1.LABEL1</TITLE>
   <LOCATION>18,36</LOCATION>
   <SIZE>94,18</SIZE>
   <HELPLINK>PANEL1.LABEL1</HELPLINK>
  </LABEL>
  <TEXTFIELD name="TEXTFIELD1">
   <TITLE>PANEL1.TEXTFIELD1</TITLE>
   <LOCATION>125,31</LOCATION>
   <SIZE>191,26</SIZE>
   <DATACLASS>SampleBean</DATACLASS>
   <ATTRIBUTE>UserData</ATTRIBUTE>
   <STRING minlength="0" maxlength="15"/>
   <HELPALIAS>LABEL1</HELPALIAS>
  </TEXTFIELD>
  <BUTTON name="BUTTON1">
   <TITLE>PANEL1.BUTTON1</TITLE>
   <LOCATION>125,100</LOCATION>
   <SIZE>100,26</SIZE>
   <STYLE>DEFAULT<</STYLE>
   <ACTION>COMMIT</ACTION>
   <HELPLINK>PANEL1.BUTTON1</HELPLINK>
  </BUTTON>
 </PANEL>

</PDML>
```

資源軟體組

與每一個 PDML 檔相關的是資源組。在本例中，可轉換的資源是儲存在 PROPERTIES 檔中，稱爲 **MyGUI.properties**。請注意，PROPERTIES 檔也含有 GUI Builder 的自定資料。

```
##Generated by GUI Builder
BUTTON_1=Close
TEXT_1=
@GenerateHelp=1
@Serialize=0
@GenerateBeans=1
LABEL_1=Enter some data:
PANEL_1.Margins=18,18,18,18,18,18
PANEL_1=Simple Example
```

JavaBean

此範例也會產生 JavaBean 物件的 Java 原始程式碼架構。下面是 **SampleBean.java** 的內容：

```
import com.ibm.as400.ui.framework.java.*;

public class SampleBean extends Object        implements DataBean
{
     private String m_sUserData;
     public String getUserData()    {
         return m_sUserData;    }

     public void setUserData(String s)    {
         m_sUserData = s;    }

        public Capabilities getCapabilities()
    {
                return null;
    }

        public void verifyChanges()
    {
    }

        public void save()
    {
    }

        public void load()
    {
         m_sUserData ="";    }
}
```

請注意，架構已含有 UserData 特性的 gettor其它的方法是由 DataBean 介面定義，所以是必要的。

GUI Builder 已對架構呼叫了 Java 編譯器，且產生了相對應的類別檔案。就此簡式範例而言，您不需要修改 Bean 實作即可達到目的。在眞正的 Java 應用程式中，您通常要修改 load 及 save 方法，才能轉送外部資料來源的資料。其他兩個方法的預設實作通常就足夠了。如需其它相關資訊，請參閱有關 **DataBean** 介面的說明文件，該文件是在 PDML 執行時間組織架構的 javadocs。

說明檔

GUI Builder 也會建立一個 HTML 組織架構，稱爲 Help Document。解說寫出器可以編輯此檔案，以輕易地管理解說資訊。若需其餘相關資訊，請參閱下列主題：

- 建立 Help Document
- 編輯由 GUI Builder 產生的 Help Document

建構應用程式

一旦您已儲存畫面定義及產生的檔案，您就可以建構應用程式了。您所需要的就是一個新的 Java 來源檔，其中含有應用程式的主進入點。就本例而言，該檔案稱爲 **SampleApplication.java**。它包含了下列程式碼：

```
import com.ibm.as400.ui.framework.java.*;
import java.awt.Frame;

public class SampleApplication{
     public static void main(String[] args)
    {
                // Instantiate the bean object that supplies data to the panel
                SampleBean bean = new SampleBean();

                // Initialize the object
                bean.load();

                // Set up to pass the bean to the panel manager
                DataBean[] beans = { bean };

        // Create the panel manager. Parameters:
                // 1. PDML file as a resource name
                // 2. Name of panel to display
                // 3. List of data objects that supply panel data
         // 4. An AWT Frame to
make the panel modal
        PanelManager pm = null;
         try { pm = new
PanelManager("MyGUI", "PANEL_1", beans, new Frame()); }                catch (DisplayManagerException e)
        {
            // Something didn't work, so display a message and exit
            e.displayUserMessage(null);
            System.exit(1);
        }

        // Display the panel - we give up control here
        pm.setVisible(true);

        // Echo the saved user data

System.out.println("SAVED USER DATA: '" + bean.getUserData() +
"'");
        // Exit the application
        System.exit(0);
    }
}
```

呼叫程式的責任在於，呼叫 **load** 來起始設定 bean 物件。 如果畫面的資料由多重 bean 物件所提供，則在傳遞每一個物件到 Graphical Toolbox 環境之前，應先起始設定它們。

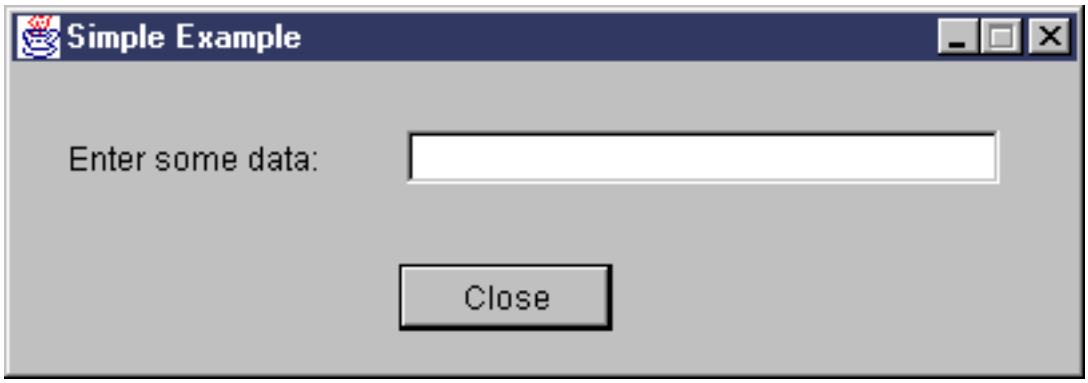
類別 **com.ibm.as400.ui.framework.java.PanelManager** 提供 API 來顯示獨立式視窗及對話框。 同於建構子上提供的名稱的 PDML 檔名會被 Graphical Toolbox 視爲資源名稱 - 目錄、ZIP 檔或含有 PDML 的 JAR 須在 classpath 中識別。

因爲 **Frame** 物件是在建構子上提供的，所以視窗將以限制模式的對話框運作。在眞正的 Java 應用程式中，這個物件可能取自於適合對話框的上層視窗。因爲視窗是限制模式的，所以控制權不會傳回到應用程式，直到使用者關閉視窗爲止。此時，應用程式僅回應修改過的使用者資料並結束執行。

執行應用程式

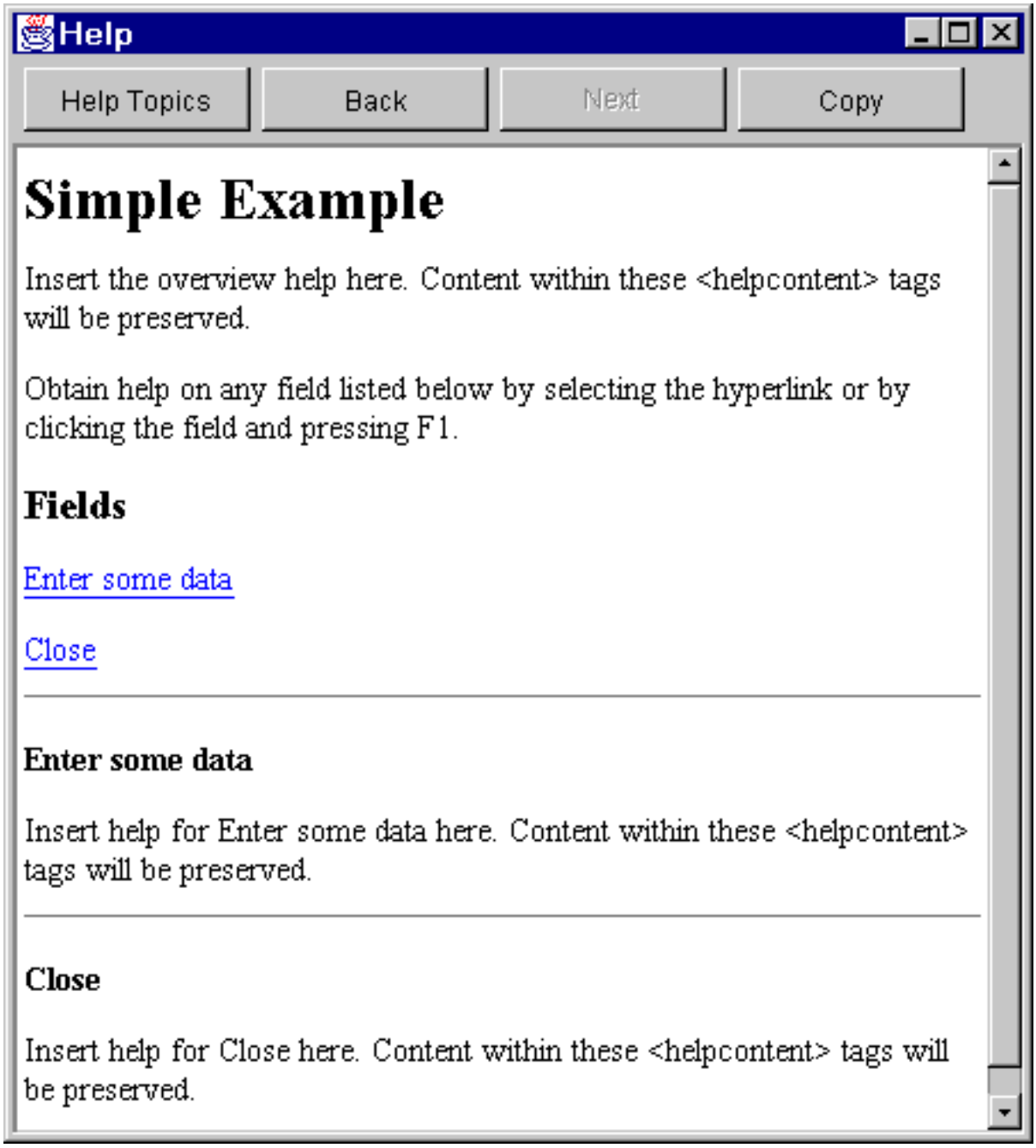
底下是編譯及執行應用程式時視窗的樣子：

圖 10：簡式範例應用程式視窗



當焦點在文字欄位上時，若使用者按下 F1，則 Graphical Toolbox 將顯示說明瀏覽器，它含有 GUI Builder 產生的線上說明架構。

圖 11：簡式範例線上說明架構



您可以編輯 HTML，並新增範例中所示的說明主題的實際說明內容。

如果文字欄位中的資料無效 (例如，如果使用者未提供一個值，即按一 下 **Close** 按鈕)，則 Graphical Toolbox 將顯示一則錯誤訊息，並傳回無點給欄位，以便可以輸入資料。

圖 12：資料錯誤訊息

有關如何執行這個範例作爲 applet 的資訊，請參閱在瀏覽器中使用 Graphical Toolbox 。

可編輯的組合框

當控制項產生器建立「可編輯的組合框」之取出元及設定元時，就預設值而言，它會傳回設定元中的「字串」並採用取出元中的字串參數。若將設定元變更爲採用「物件」類別，並將取出元變更爲傳回「物件」類型，會很有幫助。 這可讓您使用 ChoiceDescriptors 來決定使用者選項。

如果已偵測到取出元及設定元的物件 (Object) 類型，則系統會希望輸入 ChoiceDescriptor 或物件類型，而非格式化字串。

下列範例假設 Editable 是可編輯的組合框 (ComboBox)，它有可能使用雙精度浮點數值 (Double value)，或使用系統值，也可能並未設定。

```
public Object getEditable()
{
        if (m_setting == SYSTEMVALUE)
    {
            return new ChoiceDescriptor("choice1","System Value");
    }
        else if (m_setting == NOTSET)
    {
            return new ChoiceDescriptor("choice2","Value not set");
    }
    else
    {
            return m_doubleValue;
    }
}
```

同樣地，在已偵測到取出元及設定元的物件類型時，系統會傳回一物件，該物件是含由已選取選項的 ChoiceDescriptor 或一「物件」類型。

```
     public void setEditable(Object item)
     {
            if (ChoiceDescriptor.class.isAssignableForm(obj.getClass()))
         {
            if (((ChoiceDescriptor)obj).getName().equalsIgnoreCase("choice1"))
                m_setting = SYSTEMVALUE;
            else
                m_setting = NOTSET;
         }
         else if (Double.class.isAssignableFrom(obj.getClass()))
         {
            m_setting = VALUE;
            m_doubleValue = (Double)obj;
         }
         else
         { /* error processing */ }
     }
```

範例：使用 RecordListFormPane

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// RecordListFormPane example.This program presents a form that contains
// the contents of a file on the server.
//
// Command syntax:
//RecordListFormPaneExample system fileName
//
// This source is an example of IBM Toolbox for Java "RecordListFormPane".
//
///////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class RecordListFormPaneExample
{


     public static void main(String[] args)
    {
        // If a system and fileName was not specified, then display
        // help text and exit.
        if (args.length != 2)
        {
            System.out.println("Usage:  RecordListFormPaneExample system fileName");
                return;
        }

         try
        {
             // Create an AS400 object.
The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);

            // Create a frame.
            JFrame f = new JFrame ("RecordListFormPane example");

            // Create an error dialog adapter.This will display
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

            // Create a record list form pane to present the contents
            // of the database.Note we create the form pane, add
// the error listener, then set the system and file name.
            // Creating the form pane and setting its parameters
            // can be done in one step as follows:
            //RecordListFormPane formPane = new RecordListFormPane (system, args[1]);
            // The potential problem is there is no error listener yet
            // so if the file name is not correct, there is no place
            // to display the error.
            RecordListFormPane formPane = new RecordListFormPane();
            formPane.addErrorListener (errorHandler);
            formPane.setSystem(system);
            formPane.setFileName(args[1]);

            // Retrieve the information from the system.
             formPane.load ();
```

```
            // When the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
                public void windowClosing (WindowEvent event)
                {
                     System.exit(0);
                }
            });

            // Layout the frame with the form pane.
            f.getContentPane ().setLayout (new BorderLayout ());
            f.getContentPane ().add ("Center", formPane);
            f.pack ();
            f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
           System.exit (0);
        }
    }

}
```

使用 GUI Builder 建立 Pane (畫面)

使用 GUI Builder 建立畫面是一件很簡單的事。在 GUI Builder 主視窗的功能表列上，選取 **File → New File**。

在 GUI Builder 的 **File** 視窗的功能表列上，按一下 Insert New Panel 圖 示  ，顯示可於其中插入畫面元件的畫面建置器。**Panel** 視窗的工具列按鈕代表可新增至畫面的不同元件。選取您要的元件，然後在您要放置元件的位置上按一下。

下圖顯示利用數個您可使用的選項所建立的畫面。

圖 1：使用 GUI Builder 建立畫面範例

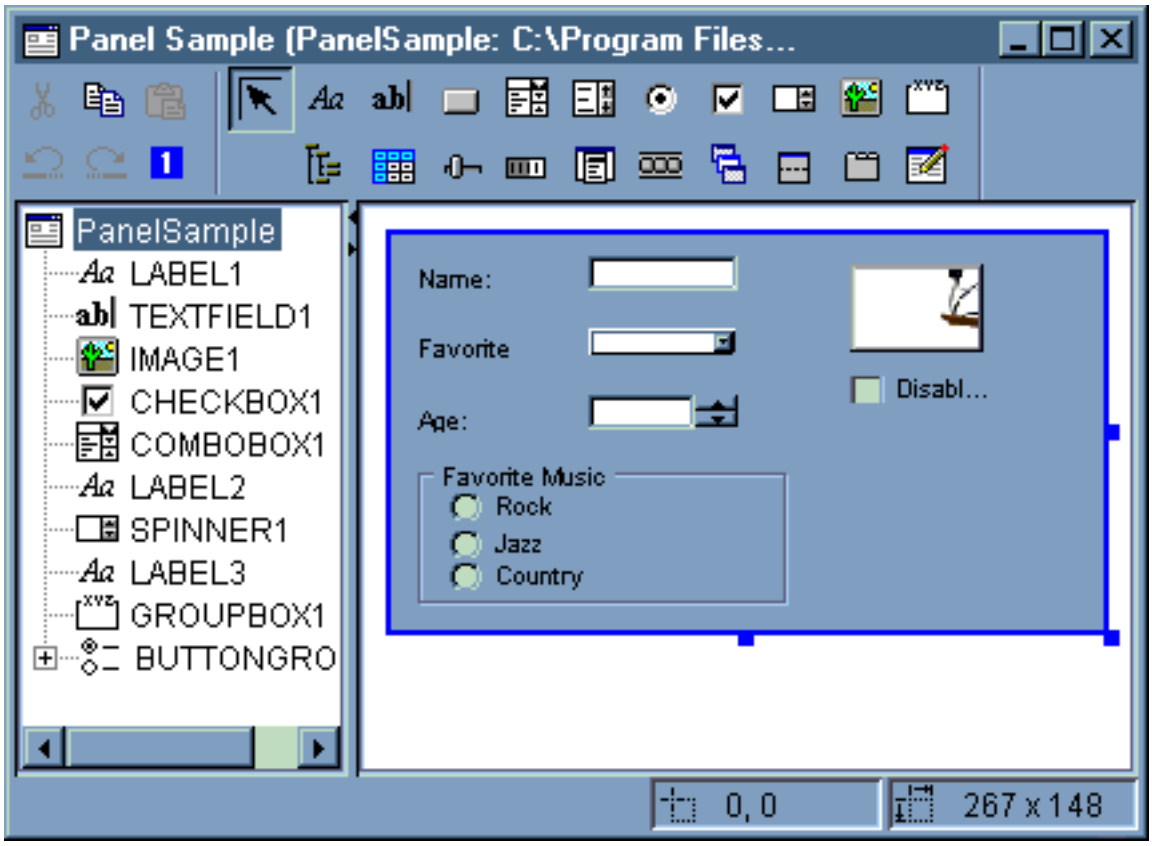


圖 1 中的畫面範例使用下列 DataBean 程式碼，將不同的元件聚集在一起：

```
import com.ibm.as400.ui.framework.java.*;

public class PanelSampleDataBean extends Object
        implements DataBean
{
        private String m_sName;
        private Object m_oFavoriteFood;
        private ChoiceDescriptor[] m_cdFavoriteFood;
        private Object m_oAge;
        private String m_sFavoriteMusic;

        public String getName()
    {
                return m_sName;
    }

        public void setName(String s)
    {
                m_sName = s;
    }

        public Object getFavoriteFood()
    {
                return m_oFavoriteFood;
    }

        public void setFavoriteFood(Object o)
    {
                m_oFavoriteFood = o;
    }

        public ChoiceDescriptor[] getFavoriteFoodChoices()
```

```
    {
            return m_cdFavoriteFood;
    }

        public Object getAge()
    {
            return m_oAge;
    }

        public void setAge(Object o)
    {
            m_oAge = o;
    }

        public String getFavoriteMusic()
    {
            return m_sFavoriteMusic;
    }

        public void setFavoriteMusic(String s)
    {
            m_sFavoriteMusic = s;
    }

        public Capabilities getCapabilities()
    {
             return null;
    }

        public void verifyChanges()
    {
    }

        public void save()
    {
            System.out.println("Name = " + m_sName);
            System.out.println("Favorite Food = " + m_oFavoriteFood);
            System.out.println("Age = " + m_oAge);
            String sMusic = "";
            if (m_sFavoriteMusic != null)
        {
            if (m_sFavoriteMusic.equals("RADIOBUTTON1"))
                sMusic = "Rock";
            else if (m_sFavoriteMusic.equals("RADIOBUTTON2"))
                sMusic = "Jazz";
            else if (m_sFavoriteMusic.equals("RADIOBUTTON3"))
                sMusic = "Country";
        }
        System.out.println("Favorite Music = " + sMusic);
    }

    public void load()
    {
        m_sName = "Sample Name";
        m_oFavoriteFood = null;
        m_cdFavoriteFood = new ChoiceDescriptor[0];
        m_oAge = new Integer(50);
        m_sFavoriteMusic = "RADIOBUTTON1";
    }
}
```

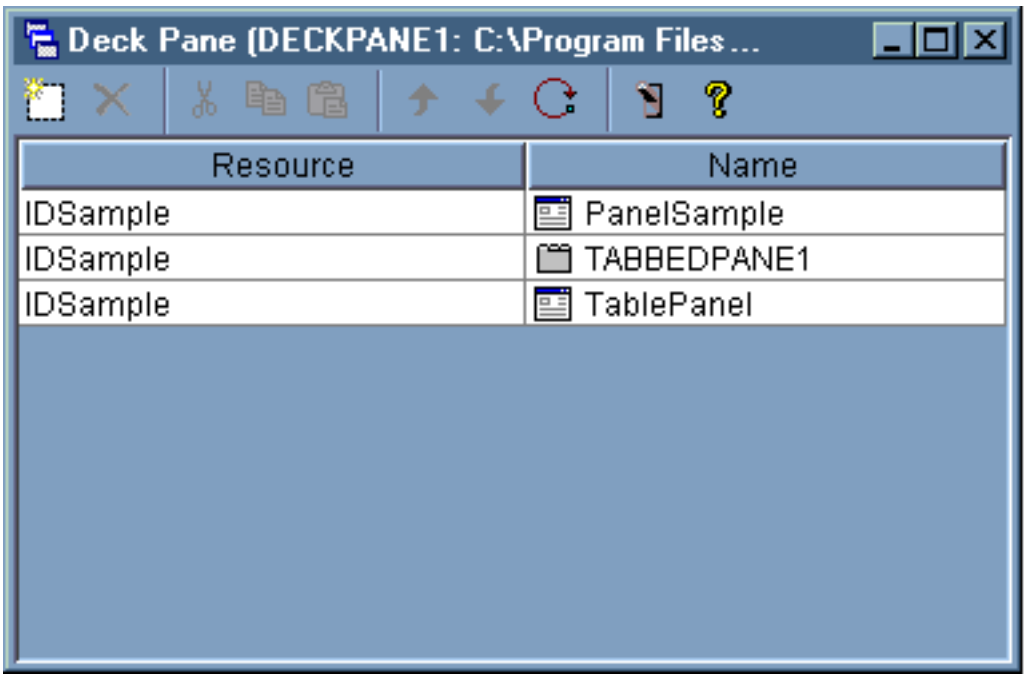
畫面是 GUI Builder 中最簡單可使用的元件，但在此簡單的畫面中，您可建置許多 UI 應用程式。

使用 GUI Builder 建立一個 Deck Pane (疊窗格)

GUI Builder 可讓建立一個疊窗格更容易。從 GUI Builder 視窗的功能表列中， 選取 **File** --> **New File**。

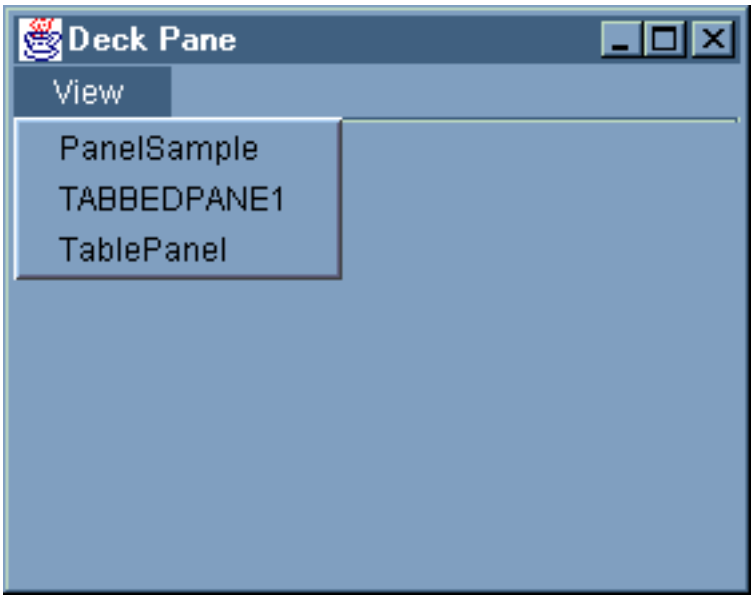
在 GUI Builder **File** 視窗的功能表列上， 按一下 **Insert Deck Pane** 工具按鈕  來顯示窗格建置器，讓您在裡面插入一個疊窗格的元件。在下列範例中，新增了三個元件。

圖 1：使用 GUI Builder 建立疊窗格



在建立一個疊窗格之後，請按一下 **Preview** 工具按鈕  來預覽它。除非您選取 **View** 功能表，否則一個疊窗格看起來是純黑白的。

圖 2：使用 GUI Builder 預覽疊窗格



從 Deck Pane **View** 功能表，選取您要檢視的元件。就此範例而言，您可以選取要檢視 PanelSample、TABBEDPANE1 或 TablePanel。下列圖示說明當您檢視這些元件時所看到的內容。

圖 3：使用 GUI Builder 檢視PanelSample

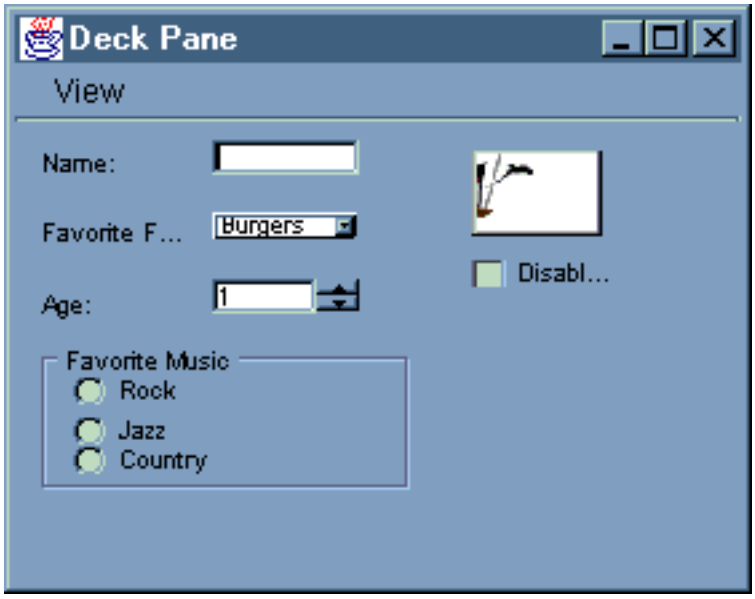


圖 4：使用 GUI Builder 檢視 TABBEDPANE1

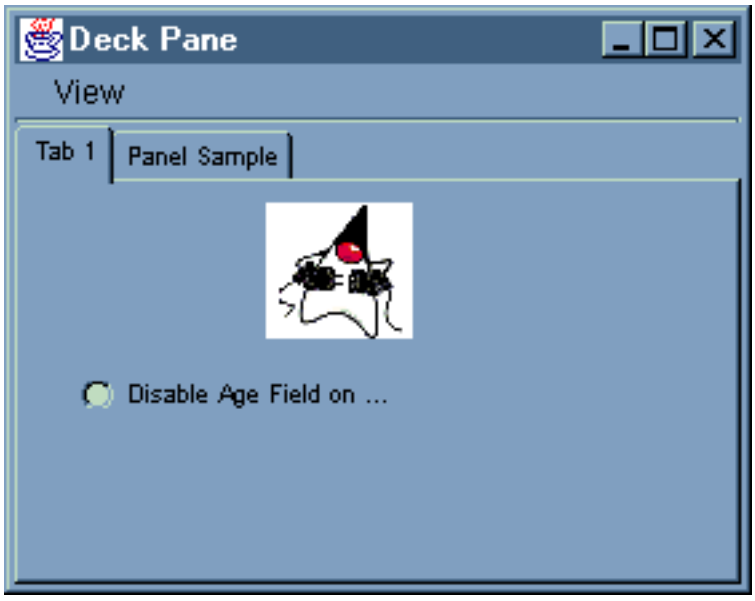
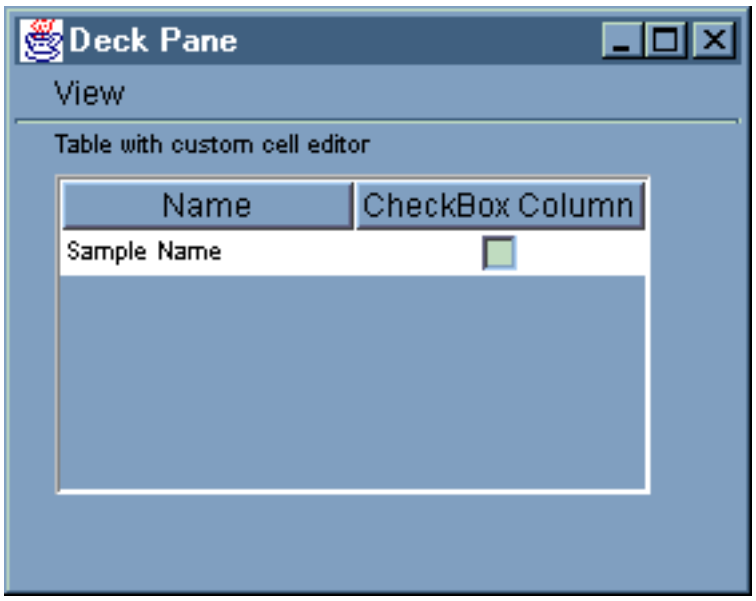


圖 5：使用 GUI Builder 檢視TablePanel

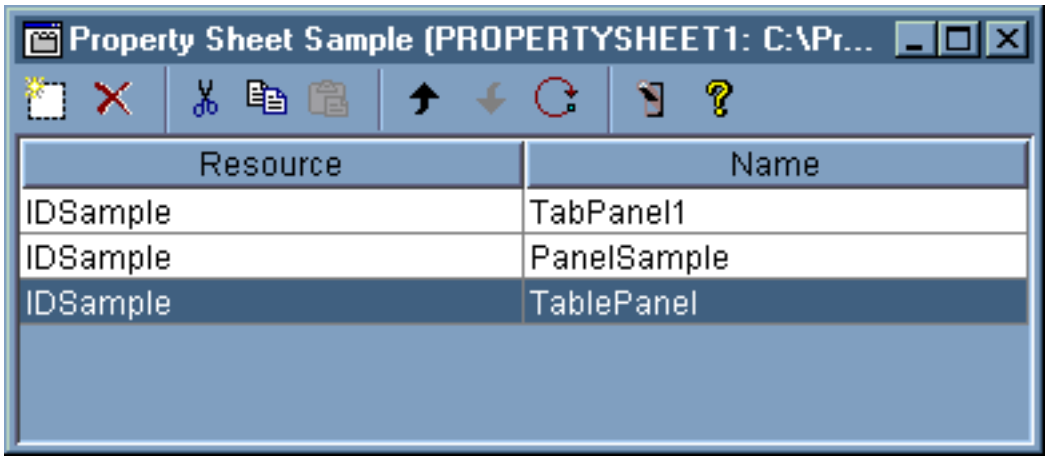


使用 GUI Builder 建立 Property Sheet (內容表)

GUI Builder 使得建立 Property Sheet 更加容易。在 GUI Builder 主視窗的功能表列上，選取 **File** --> **New File**。

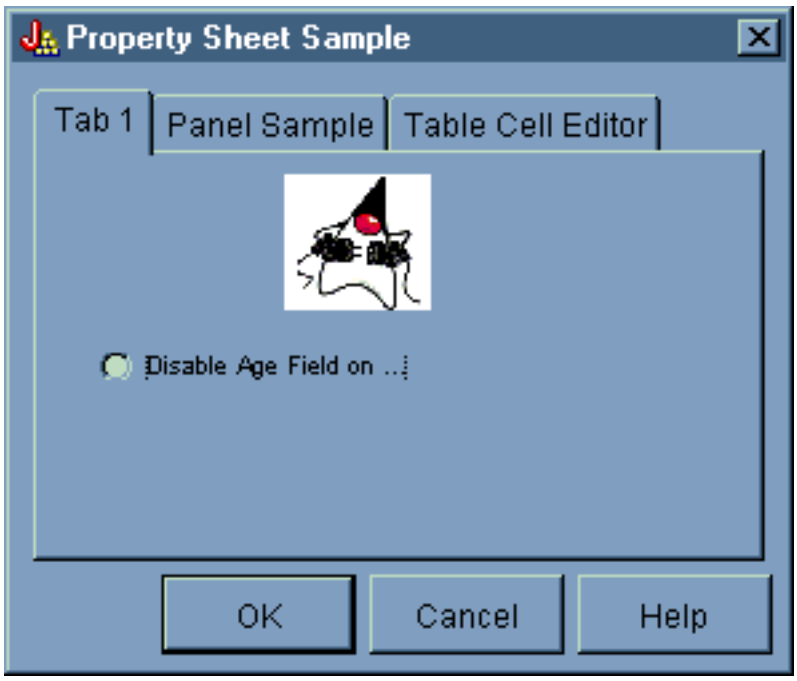
在 GUI Builder 的 **File**視窗的功能表列上， 按一下 Insert Property Sheet 圖示  即顯示出畫面建置器，您可在其中插入內容表的元件。

圖 1：使用 GUI Builder 建立 Property Sheet



建立 Property Sheet 之後，使用  圖示便可以預覽它。就此例而言，您可從三個標籤中選擇：

圖 2：使用 GUI Builder 預覽 Property Sheet

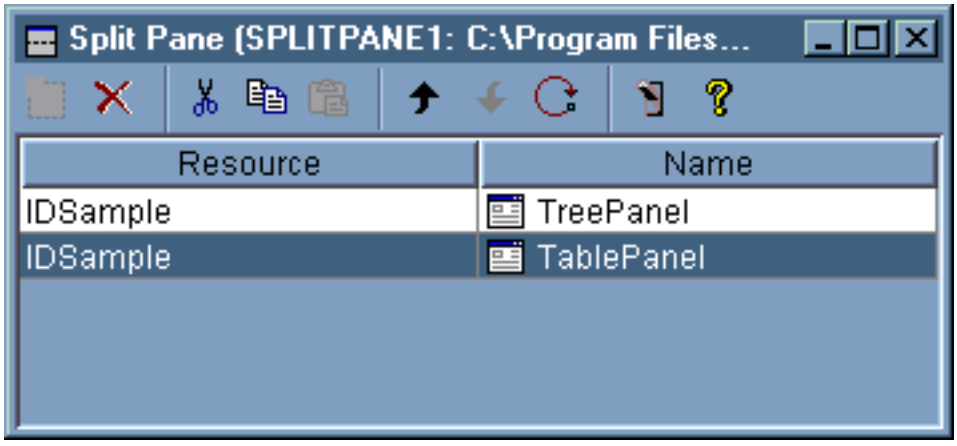


使用 GUI Builder 建立 Spilt Pane (分隔視窗)

GUI Builder 可以讓您輕鬆地建立 Spilt Pane。在 GUI Builder 主視窗的功能表列上，選取 **File** --> **New File**。

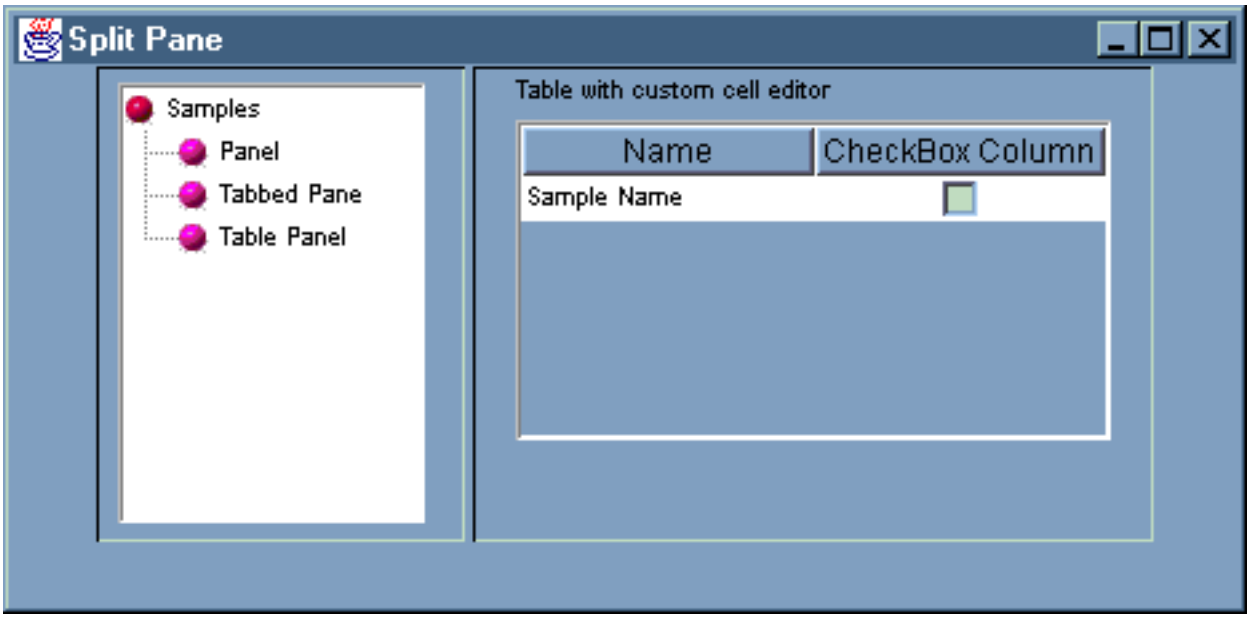
在 GUI Builder 的 **File** 視窗的功能表列上，按一下 Insert Split Pane 工具按鈕  ，以顯示畫面建置器，您就可以將想要的元件插入分割窗格中。在下列範例中，會新增兩個元件。

圖 1：使用 GUI Builder 建立 Split Pane



建立了分割窗格後，請按一下 **Preview** 工具按鈕  圖示， 來預覽 Split Pane，如圖 2 中所示。

圖 2：使用 GUI Builder 預覽 Split Pane

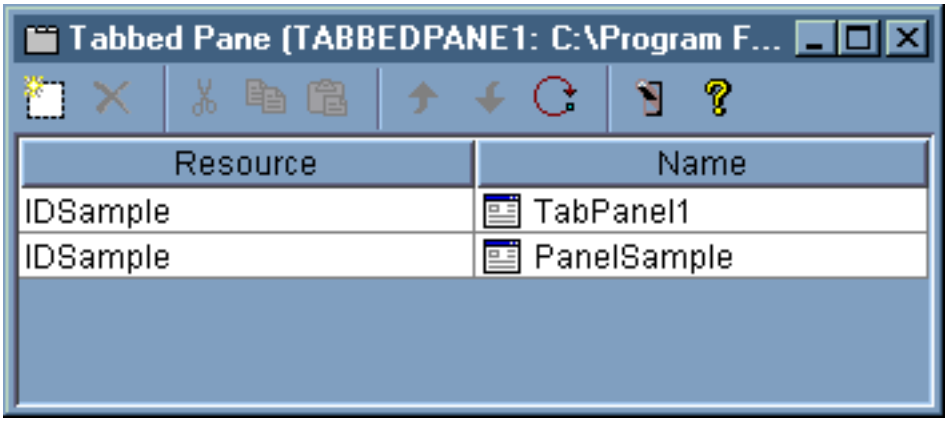


使用 GUI Builder 建立 Tabbed Pane (含標籤窗格)

GUI Builder 可以讓您輕鬆地建立一個含標籤窗格。在 GUI Builder 主視窗的功能表列上， 選取 **File** --> **New File**。

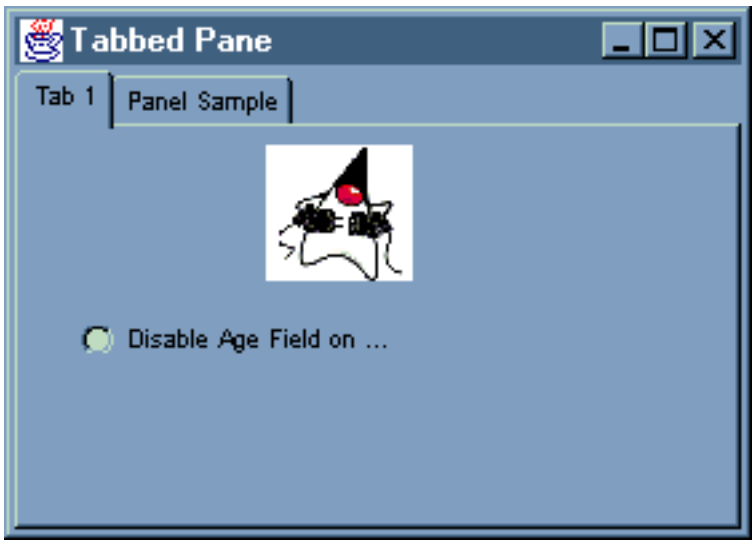
在 GUI Builder 的 **File**視窗的功能表列上， 按一下 Insert Tabbed Pane 圖示  ，以顯示畫面建置器，您就可以將想要的元件插入含標籤窗格中。在下列範例中，會新增兩個元件。

圖 1：使用 GUI Builder 建立 Tabbed Pane



建立了含標籤窗格後，請按一下 **Preview**工具按鈕  來預覽含標籤窗格。

圖 2：使用 GUI Builder 預覽 Tabbed Pane



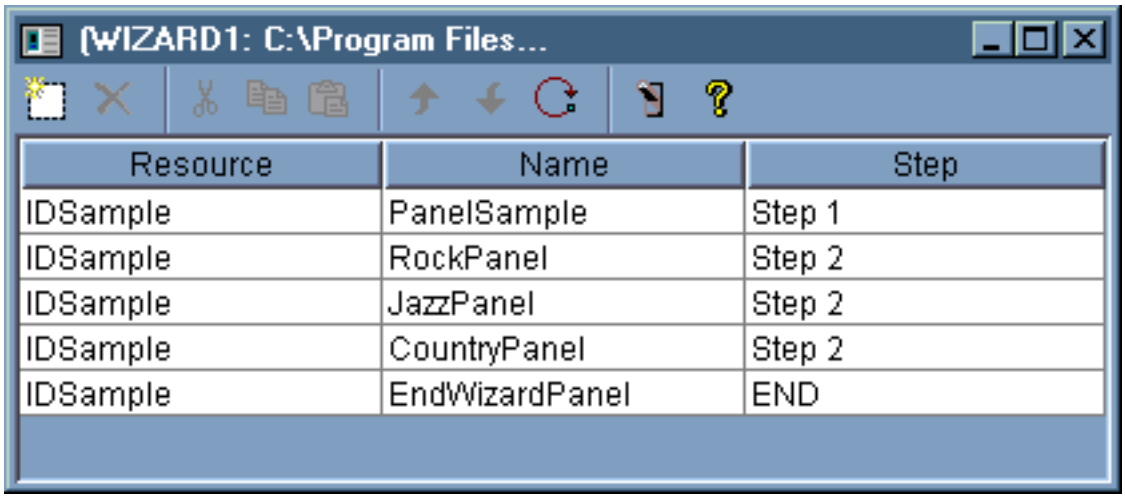
使用 GUI Builder 建立 Wizard (精靈)

GUI Builder 讓建立精靈介面變得簡單無比。從 GUI Builder 視窗的功能表列上，選取 **File** --> **New File**。

在 GUI Builder 的 **File** 視窗的功能表列上，按一下 Insert Wizard 工具列按鈕

來顯示畫面建置器，您可以在該建置器上將畫面新增到精靈中。

圖 1：以 GUI Builder 建立 Wizard



建立好精靈之後，請使用 **Preview** 工具按鈕

來預覽它。圖 2 顯示了這個範例中所出現的第一個畫面。

圖 2：以 GUI Builder 預覽第一個精靈畫面

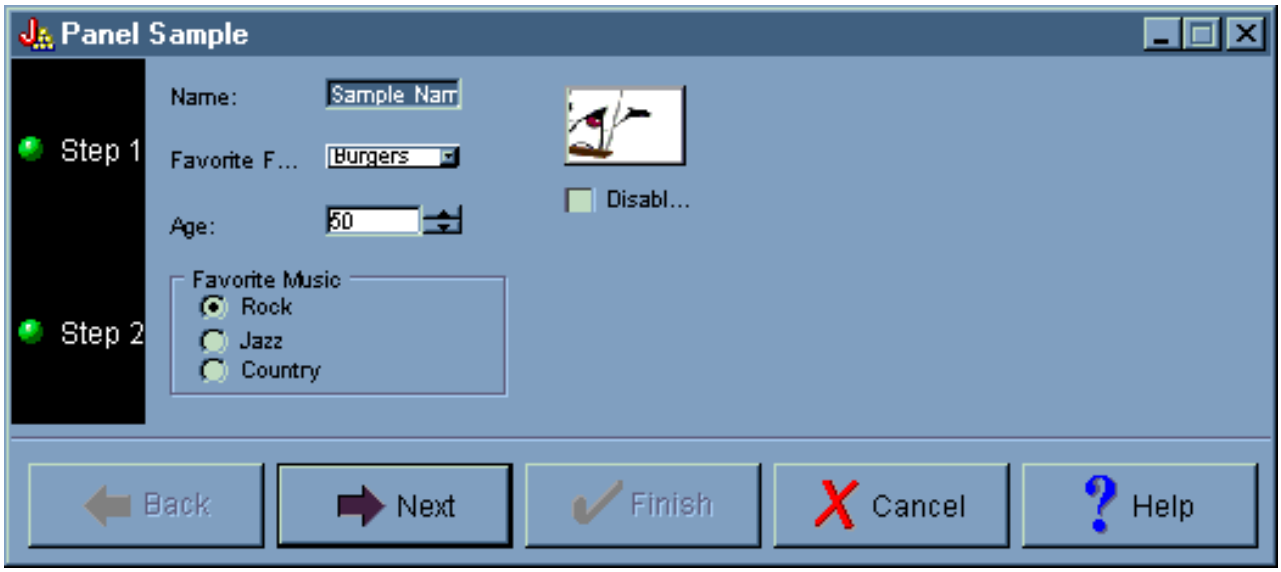
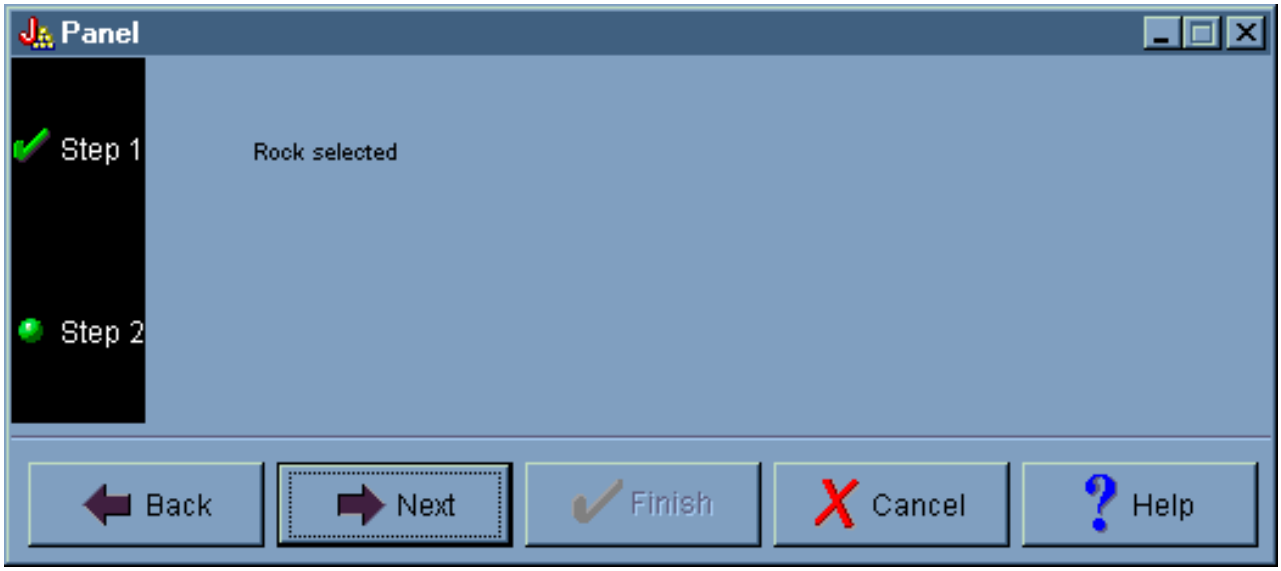


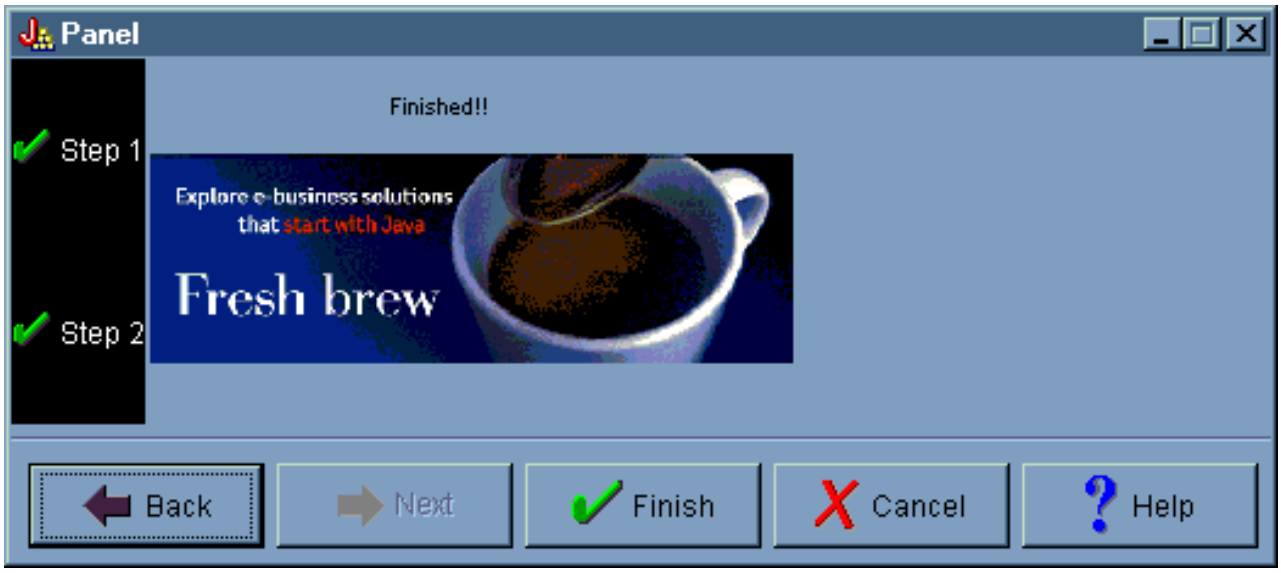
圖 2 顯示了當使用者選取 **Rock** 並按一下 **Next**時，所顯示的第二個畫面。

圖 3：以 GUI Builder 預覽第二個精靈畫面



在第二個精靈畫面按 **Next**來顯示最後一個精靈畫面，如「圖 4」所示。

圖 4：以 GUI Builder 預覽最後一個精靈畫面

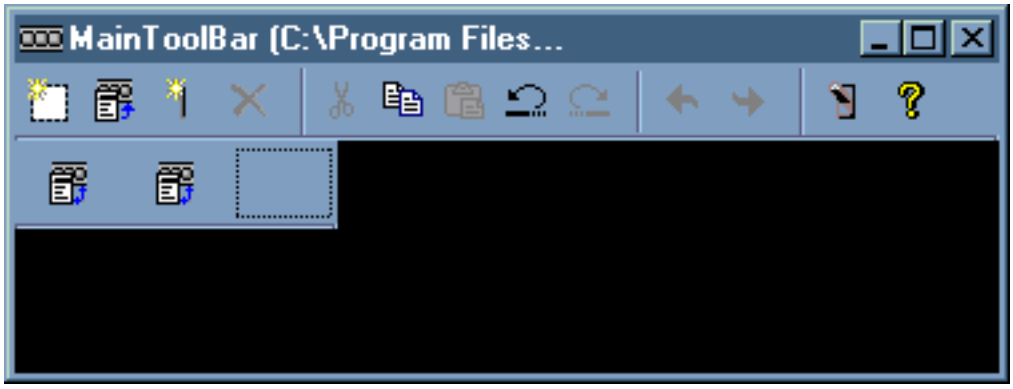


使用 GUI Builder 建立 Toolbar (工具列)

GUI Builder 可以讓您簡單地建立一個工具列。從 GUI Builder 視窗的功能表列上， 選取 **File** --> **New File**。

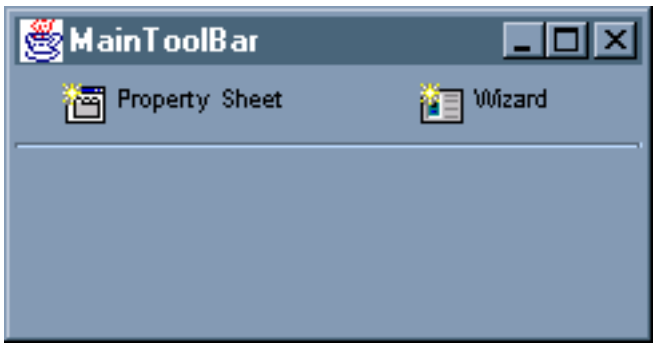
在 GUI Builder 的 **File** 視窗的功能表列上，按一下 **Insert Tool Bar**工具按鈕，以顯示畫面建置器，您就可以將想要的元件插入工具列中。

圖 1：使用 GUI Builder 建立 Tool Bar (工具列)



建立好工具列後，請按一下 **Preview**工具按鈕  來預覽工具列。就此範例而言，您可以選擇要顯示 Propery Sheet 或 Wizard。

圖 2：使用 GUI Builder 預覽 Tool Bar (工具列)

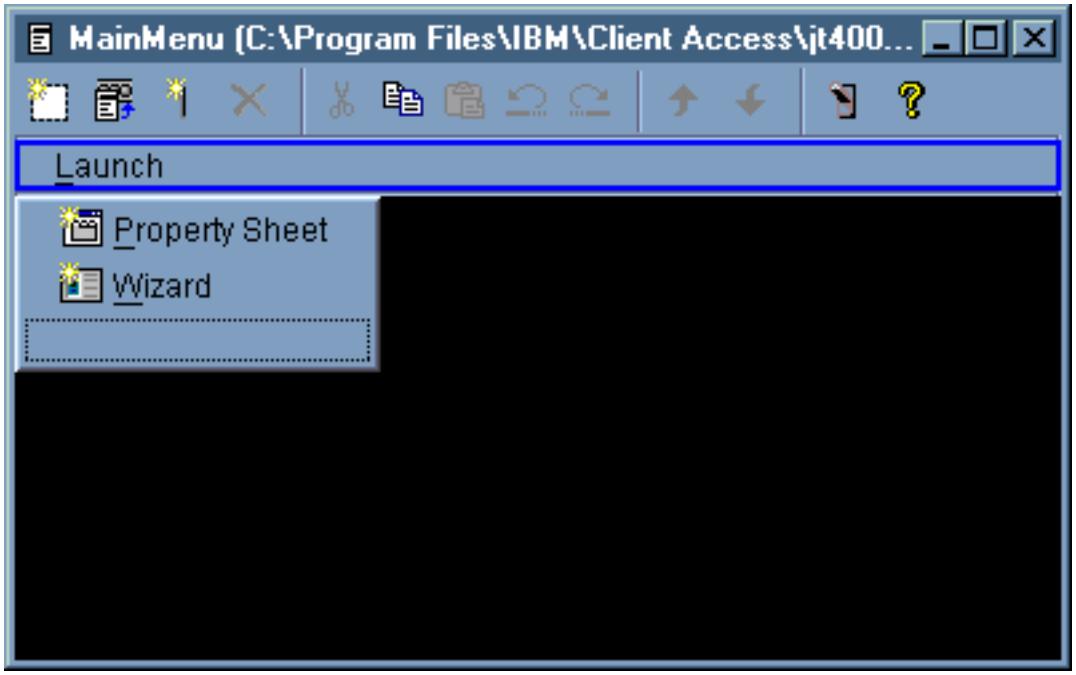


使用 GUI Builder 建立功能表列

GUI Builder 可使功能表列的建立方便又容易。從 GUI Builder 視窗的功能表列中， 選取 **File --> New File**。

在 GUI Builder **File**視窗的工具列中，按一下 **Insert Menu** 工具按鈕，建立可於其中插入功能表元件的畫面建置器。

圖 1：GUI 建置器：建立功能表



功能表建立後，使用 **Preview** 工具按鈕 加以預覽。針對此範例，您可從新建立的 **Launch** 功能表選取 **Property Sheet** 或 **Wizard**。下圖說明選取這些功能表項目時您會看見的情況。

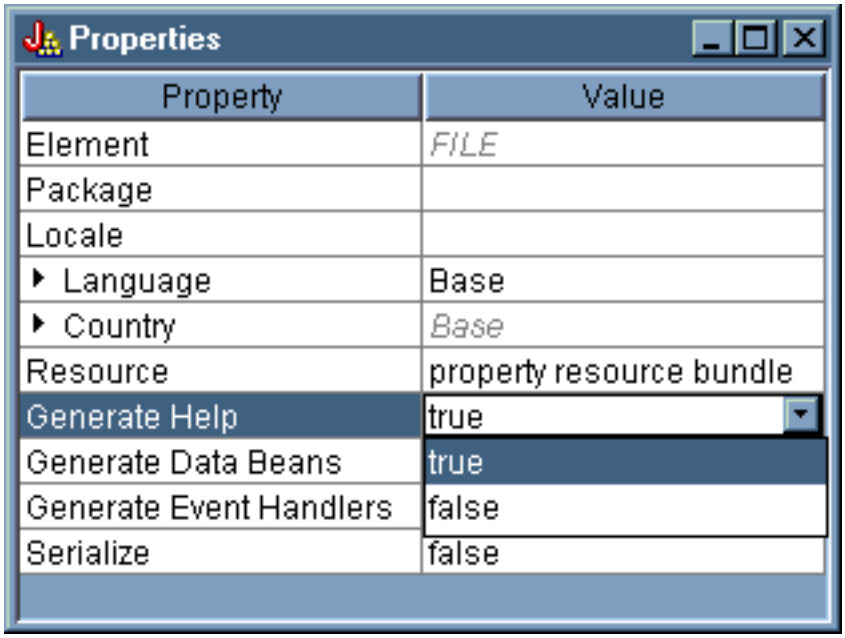
圖 2：GUI 建置器：檢視 Launch 功能表上的 Property Sheet

圖 3：GUI 建置器：檢視 Launch 功能表上的 Wizard

範例：建立 Help Document (說明檔)

使用 GUI Builder 建立說明檔是很簡單的事。在您使用的檔案內容畫面中，將 Generate help 設定爲 true。

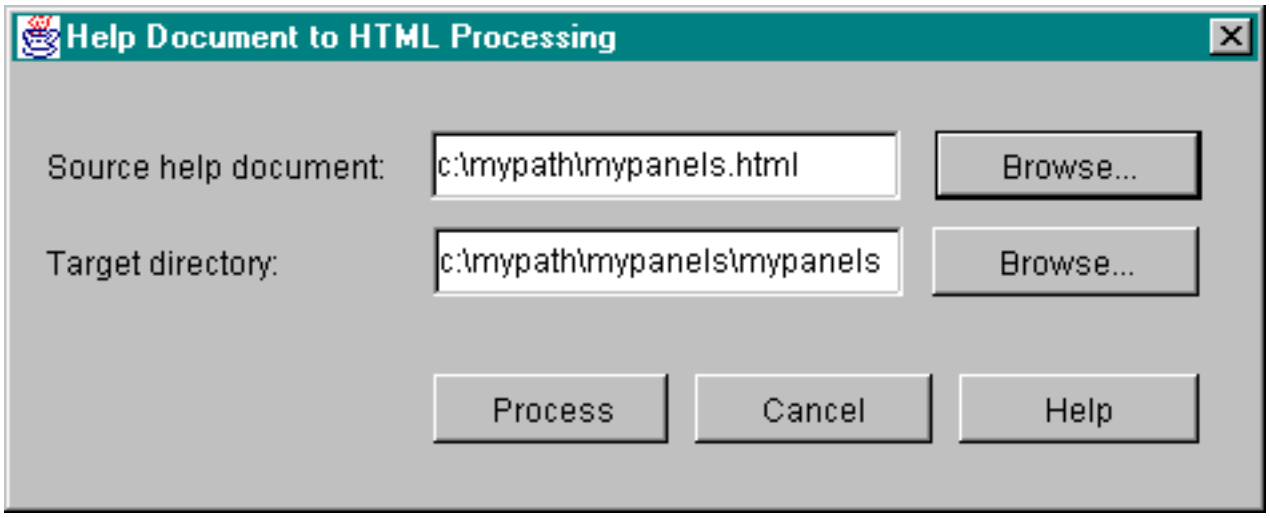
圖 1：在 GUI Builder Properties 畫面上設定 Generate Help 內容



GUI Builder 會建立一個稱爲 Help Document 的 HTML 組織架構，您可以將它加以編輯。

PDML 檔中的各個主題必須分散至不同的 HTML 檔中，才能在執行時使用。當您執行 **Help Document to HTML Processing** 時，主題會分成個別的檔案並放入在 Help Document執行環境希望個別 的 HTML 檔是在一個與 Help Document 及 PDML 檔具有相同名稱的子目錄中。**Help Document to HTML Processing** 對話會收集必要的資訊，並呼叫 HelpDocSplitter 程式以執行處理：

圖 2：Help Document to HTML Processing 對話框



在指令提示中鍵入下列，即會啓動 Help Document to HTML Processing：

```
jre com.ibm.as400.ui.tools.hdoc2htmViewer
```

執行此指令需要您的類別路徑設定正確。

如欲使用 Help Document to HTML Processing， 您要先選取具有與 PDML 檔相同名稱的 Help Document。其次，您要指定一個子目錄，其名稱與 Help Document 及 PDML 檔 相同，以供輸出之用。 選取 Process 以完成處理程序。

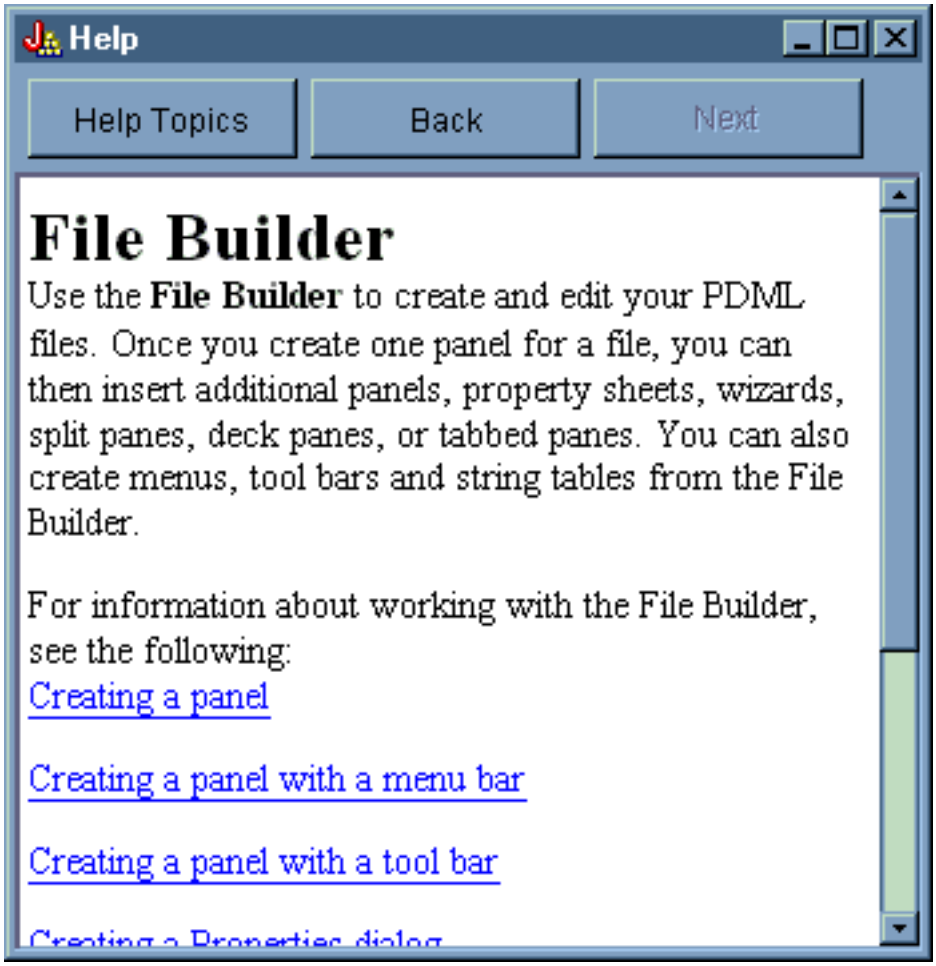
您可以在指令行中利用下列指令來分割說明文件：

```
jre com.ibm.as400.ui.tools.HelpDocSplitter "helpdocument.htm" [output directory]
```

此指令會執行分隔檔案的處理程序。您提供作爲輸入使用的 Help Document 名稱，以及選用的輸出目錄。就預設值而言，會建立與 Help Document 相同名稱的子目錄，且結果檔會放入該目錄中。

這是您看到的說明檔的範例：

圖 3：GUI Builder 說明檔範例

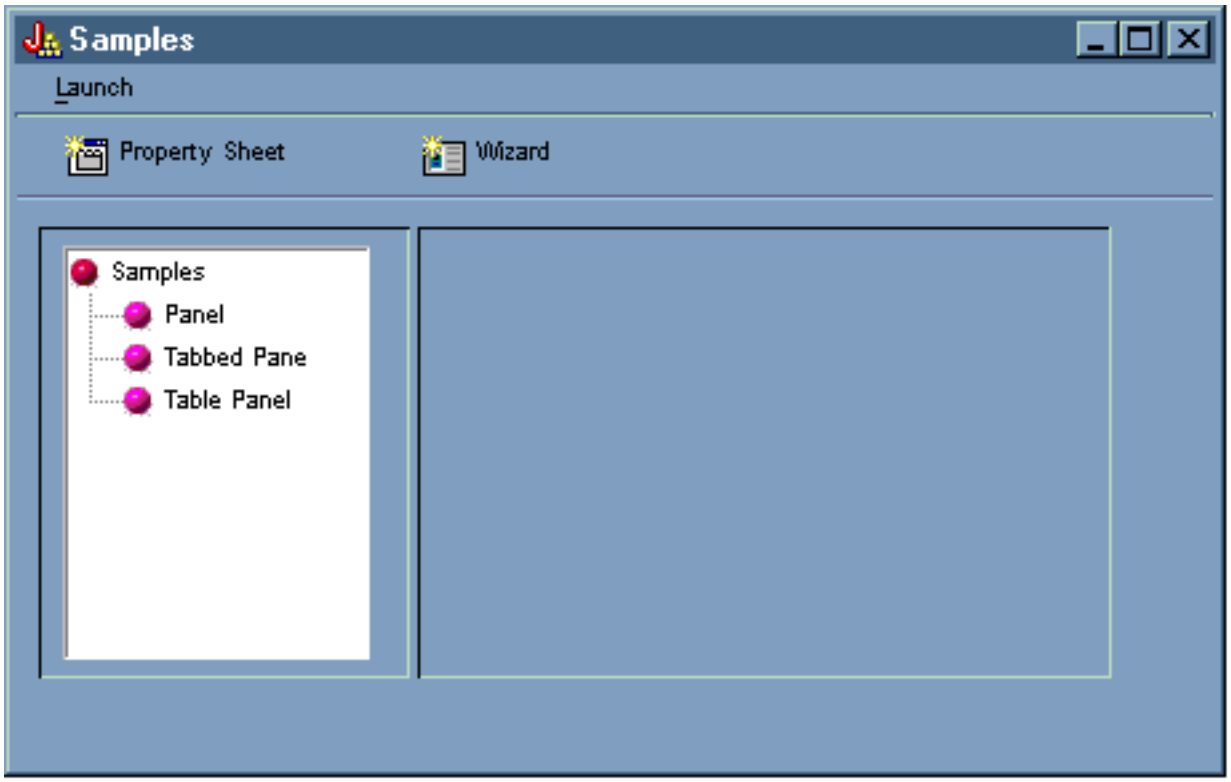


範例：使用 GUI Builder

當本區段所包含的範例與幕後運作的正確資料 bean 放在一起時， 您會取得一個完整的 GUI 應用程式。

圖 1 顯示出當您執行此範例時所顯示的第一個畫面。

圖 1：GUI Builder 範例主視窗



請注意這個螢幕可讓您使用動態畫面管理程式。圖 2 與圖 3 顯示您可以將視窗調整成較大或較小。

圖 2：調整 GUI Builder 範例主視窗的大小 (較大)

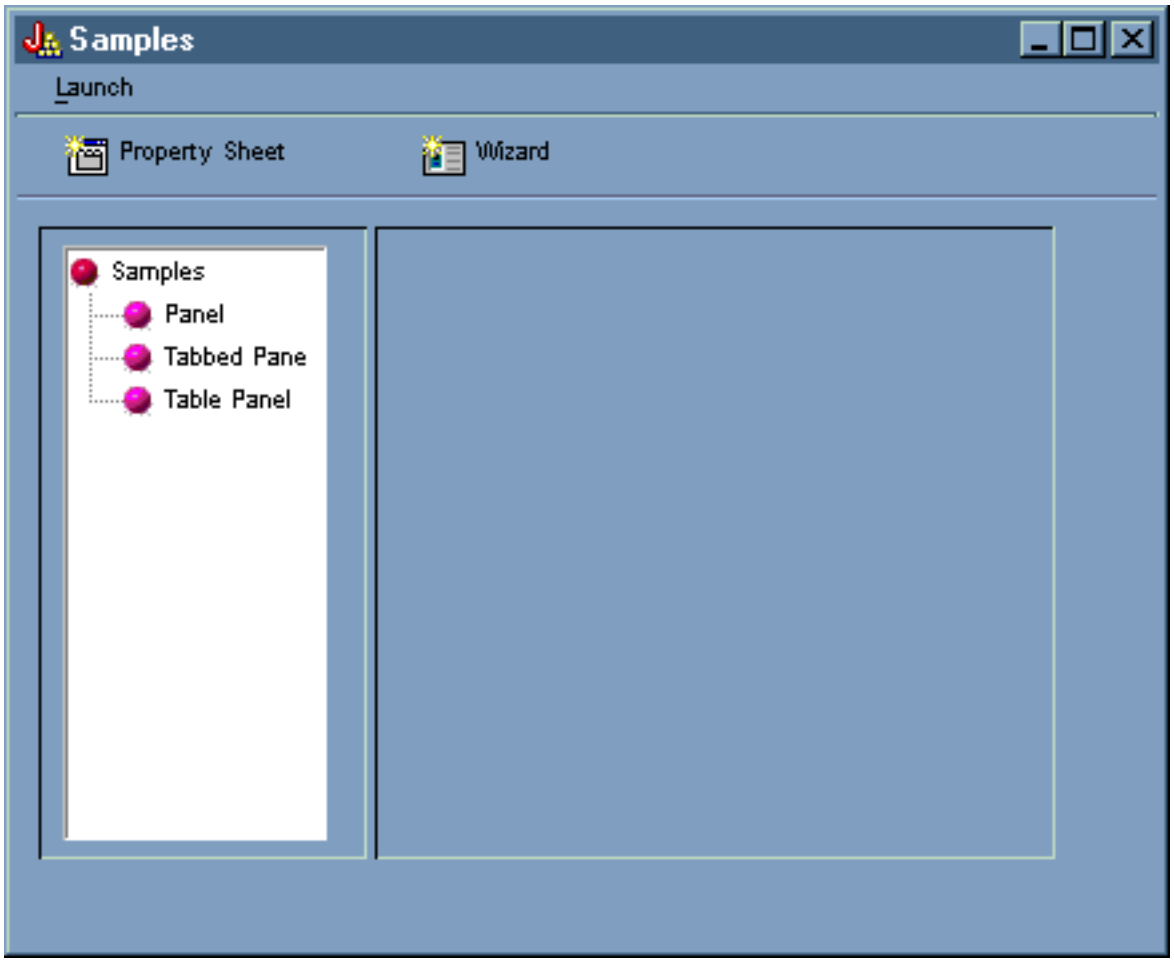
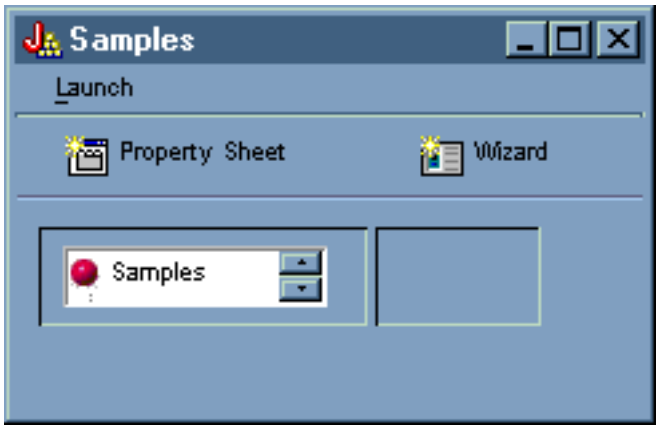


圖 3：調整 GUI Builder 範例主視窗的大小 (較小)



使用動態畫面管理程式時，文字大小並不隨著畫面以及畫面控制項大小變更。

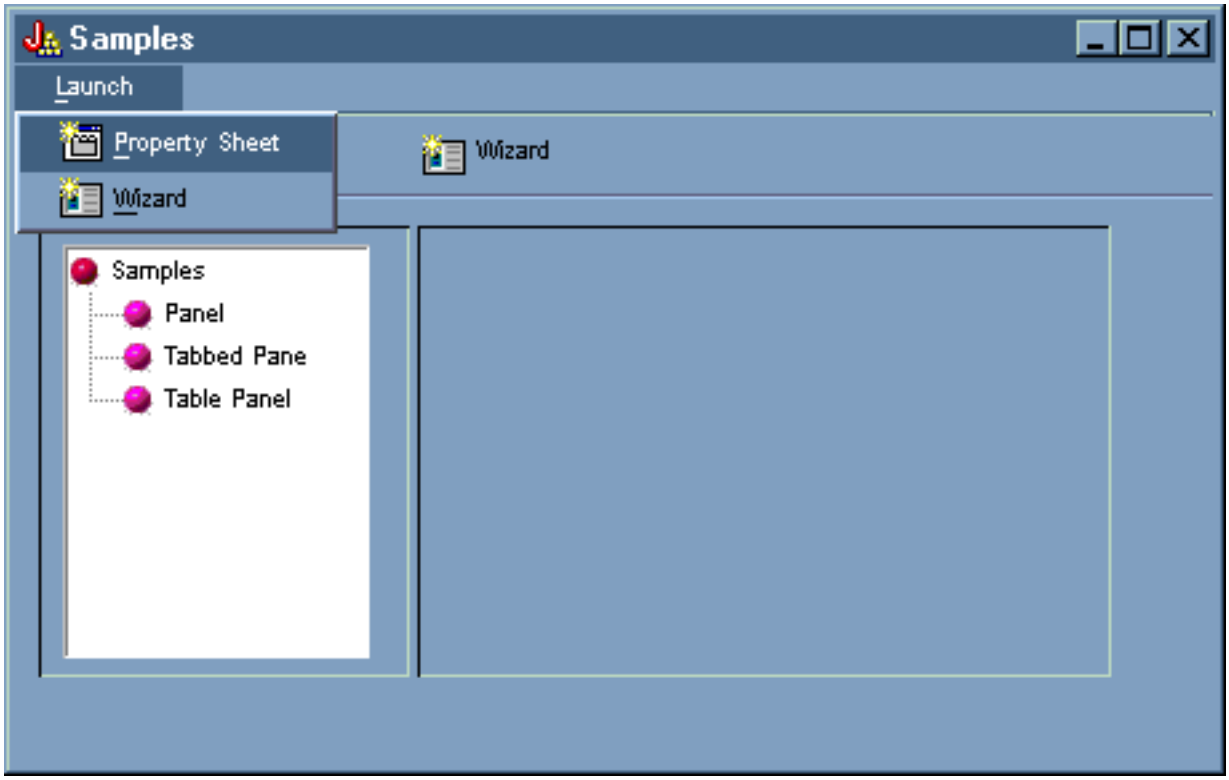
此畫面可讓您執行下列動作：

- 啓動 Property Sheet (內容表)
- 啓動 Wizard (精靈)
- 顯示左窗格中所列的範例

啓動 Property Sheet (內容表)

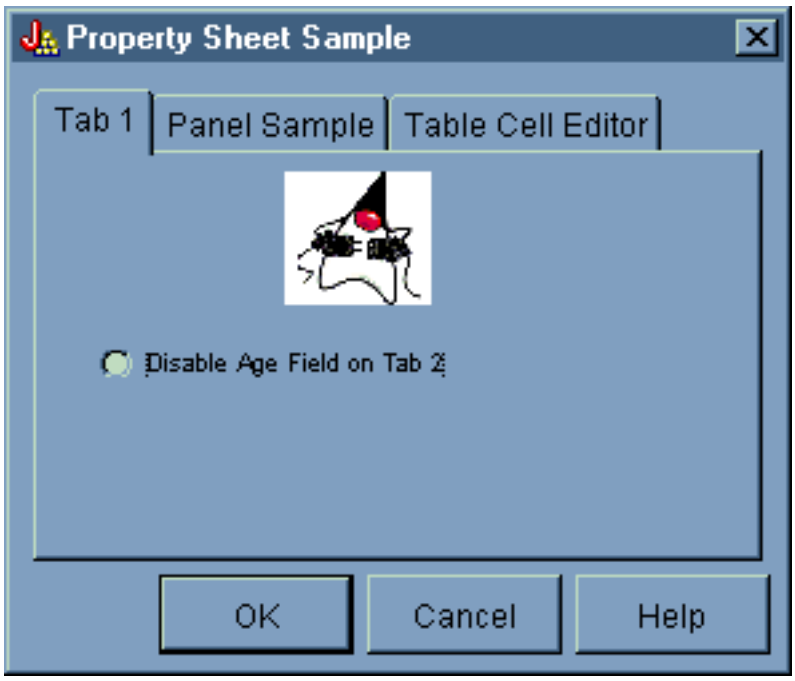
若要啓動內容表，可以按一下 Property Sheet (內容表) 工具列按鈕，或使用 **Launch** 功能表。可以在工具列和功能表之間做選擇代表功能表項目之間的鏈接。圖 4 顯示從 GUI Builder 範例主視窗中之 **Launch** 功能表選取的 **Property Sheet**。

圖 4：從 Launch 功能表選取 Property Sheet



選取 **Property Sheet** 會顯示圖 5 中的畫面。

圖 5：Property Sheet Sample 對話框



Property Sheet Sample 第一次出現時，在預設的情況下會顯示 Tab 1。圖 6 與圖 7 顯示出當您選取其它標籤時，畫面顯示的變更情形。

圖 6：選取 Panel Sample 標籤

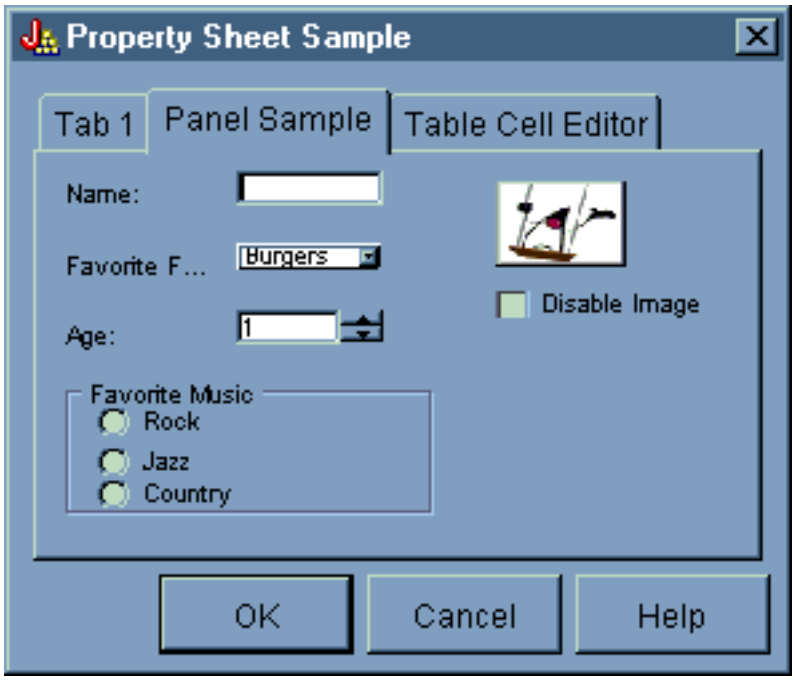
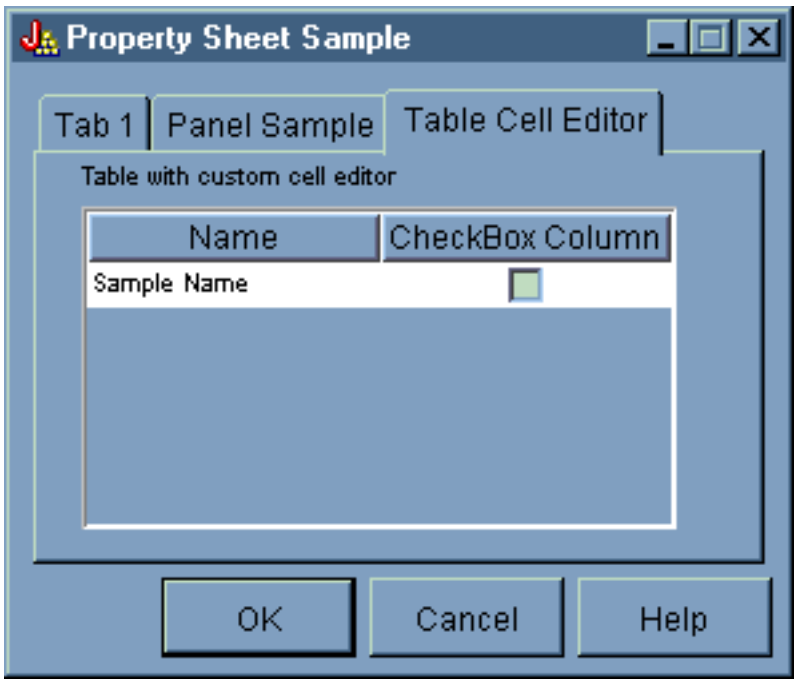


圖 7：選取 Table Cell Editor 標籤



啓動 Wizard (精靈)

若要啓動精靈，可以按一下 Wizard (精靈) 工具列按鈕或使用 **Launch** 功能表。可以在工具列和功能表之間做選擇代表功能表項目之間的鏈接。圖 8 顯示從 GUI Builder 範例主視窗中之 **Launch** 功能表選取的 **Wizard**。

圖 8：從 Launch 功能表選取 Wizard

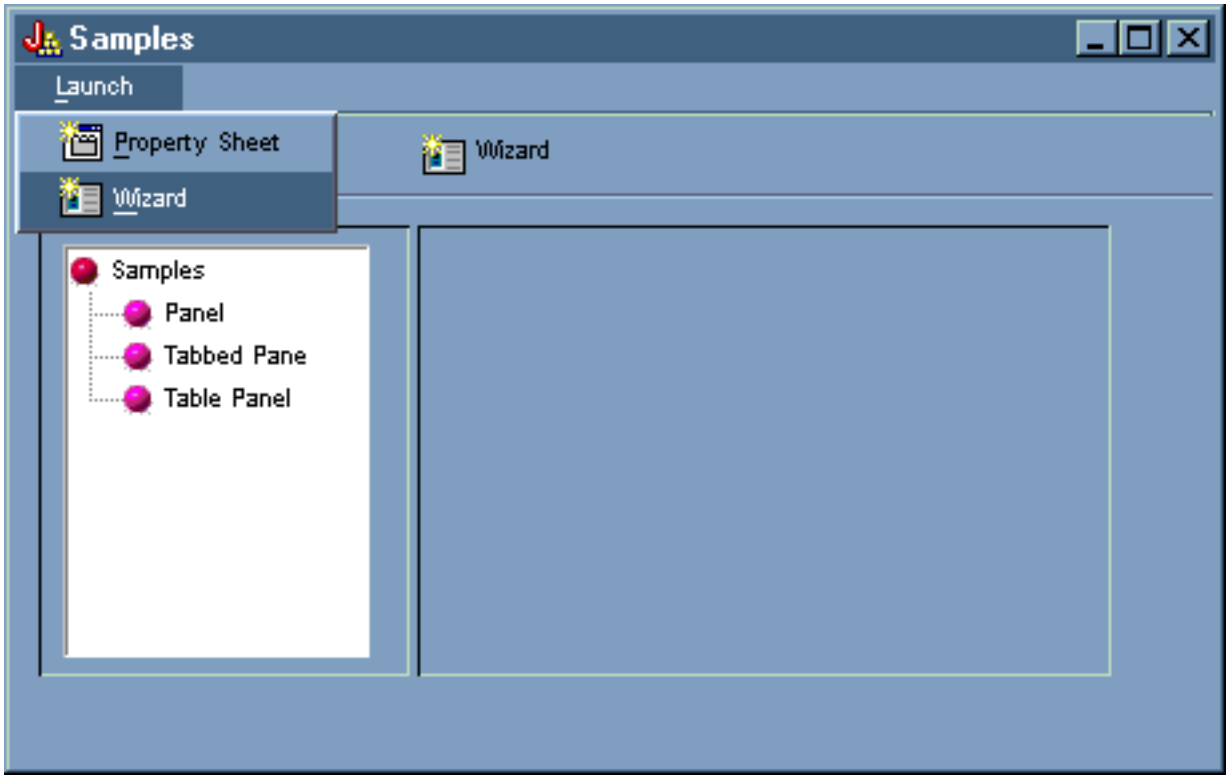
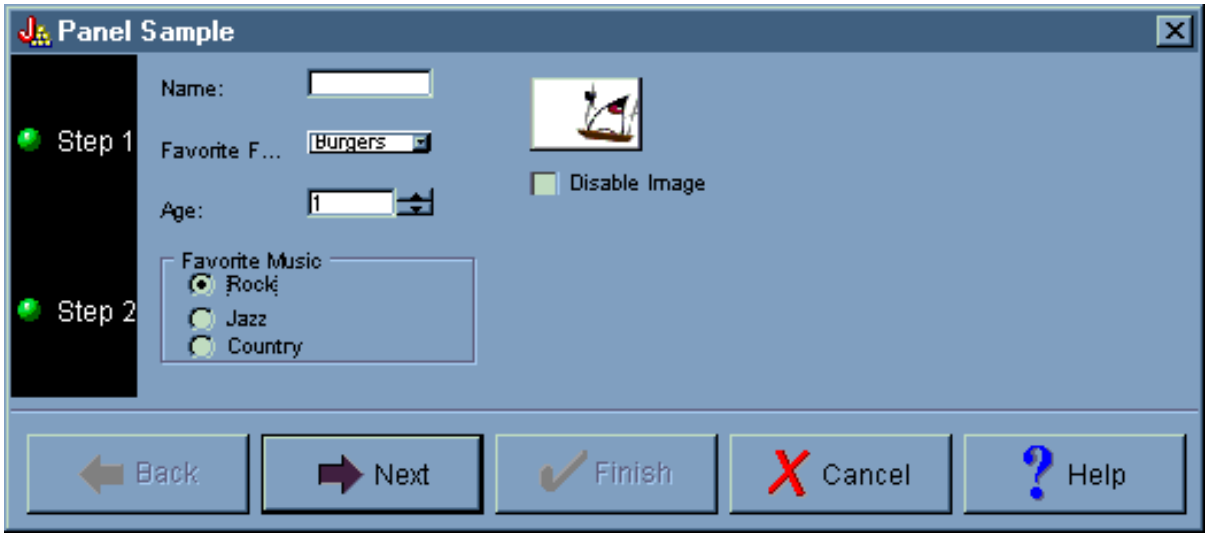


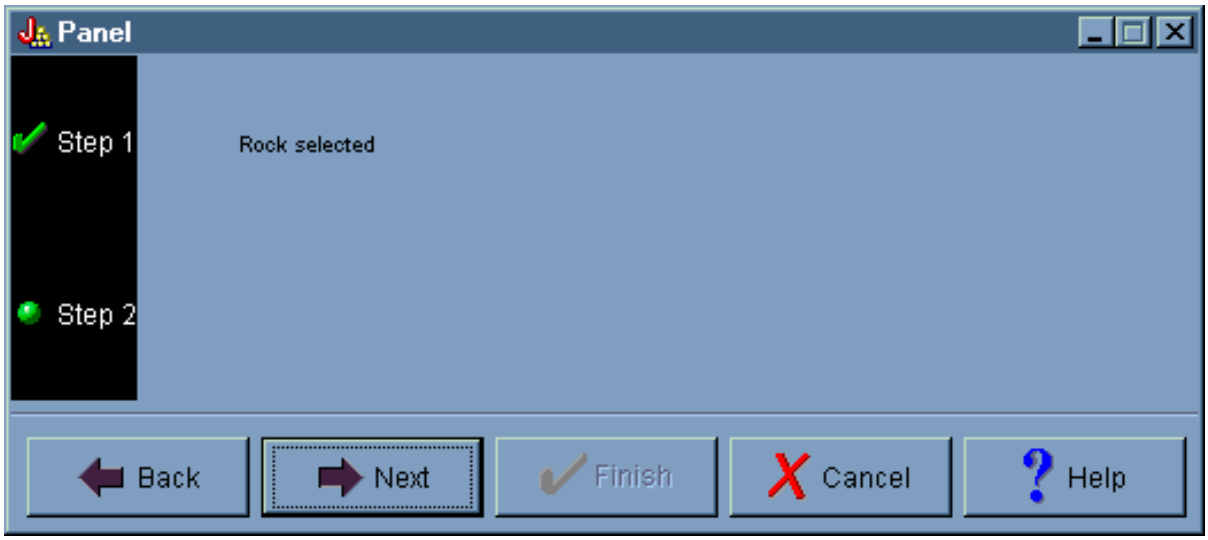
圖 9 顯示第一個精靈對話框提供的許多選項。

圖 9：選取第一個精靈對話框中的 Rock



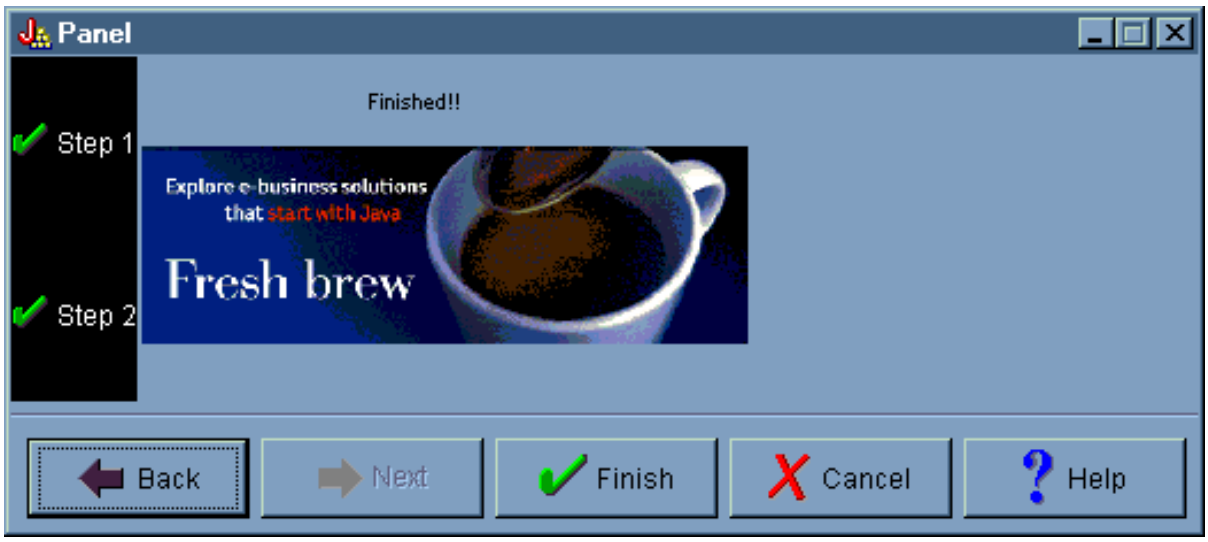
在第一個精靈對話框中，選取 **Rock** 並按一下 **Next**， 以顯示如圖 10 中顯示的第二個精靈對話框。

圖 10：第二個精靈對話框 (選取了 Rock 之後)



在第二個精靈對話框中，按一下 **Next**，以顯示如圖 11 中顯示的最後一個精靈對話框。

圖 11：最後一個精靈對話框



然而，此範例程式已被設計具迴圈功能。選取第一個精靈對話框 (圖 12) 中的 **Country**，然後按一下 **Next**，以顯示第二個精靈對話框 (圖 13)。按一下第二個精靈對話框中的 Next，會遞迴至顯示第一個對話框 (圖 14)，而不是顯示最後一個精靈對話框。

圖 12：選取第一個精靈對話框中的 Country

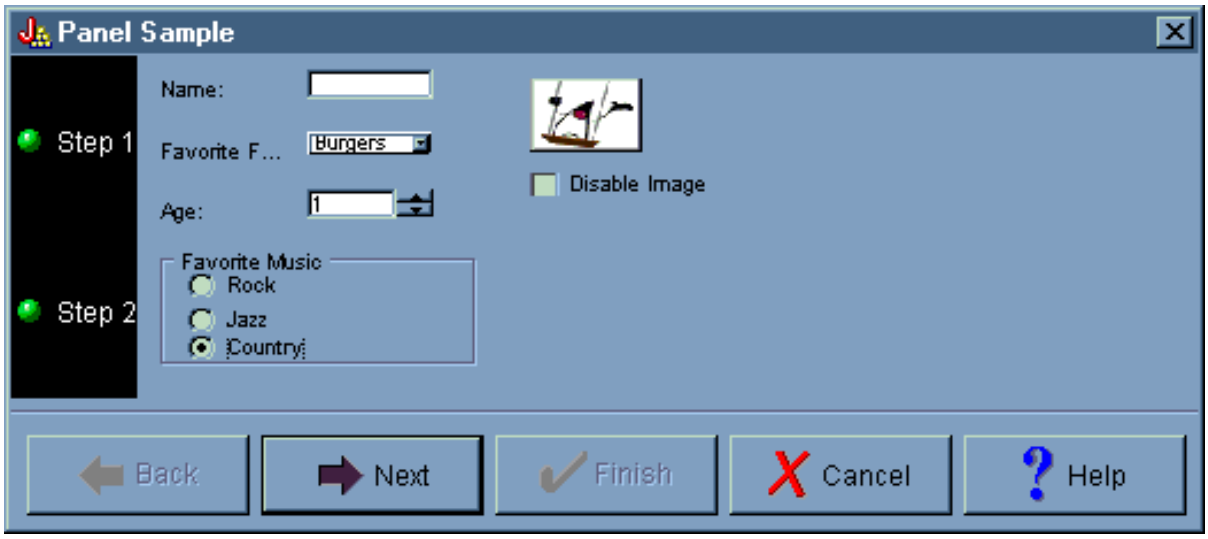


圖 13：第二個精靈對話框 (選取了 Country 之後)

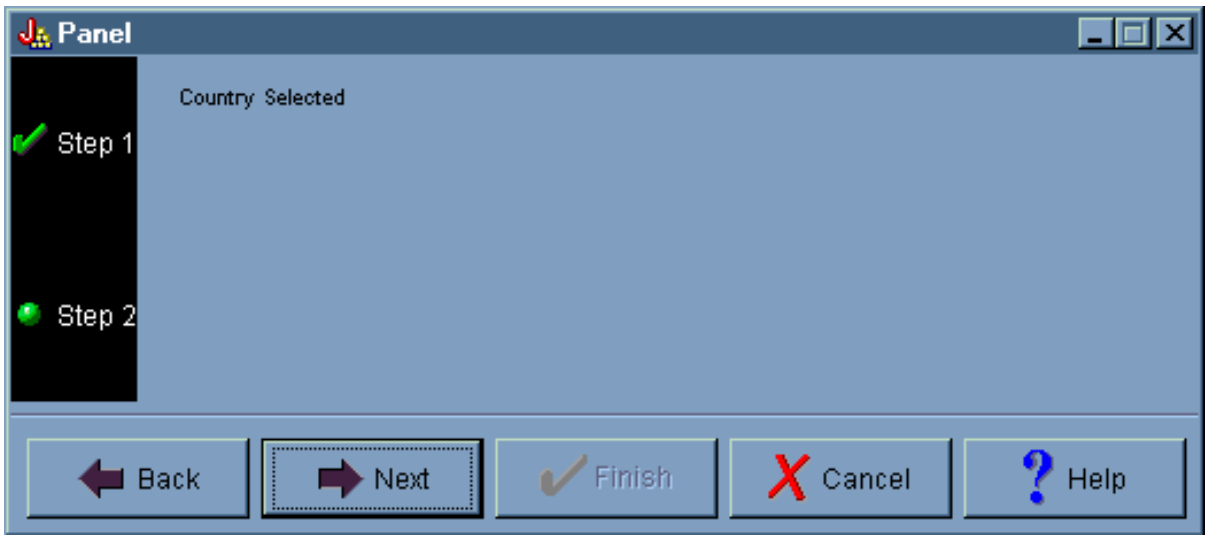
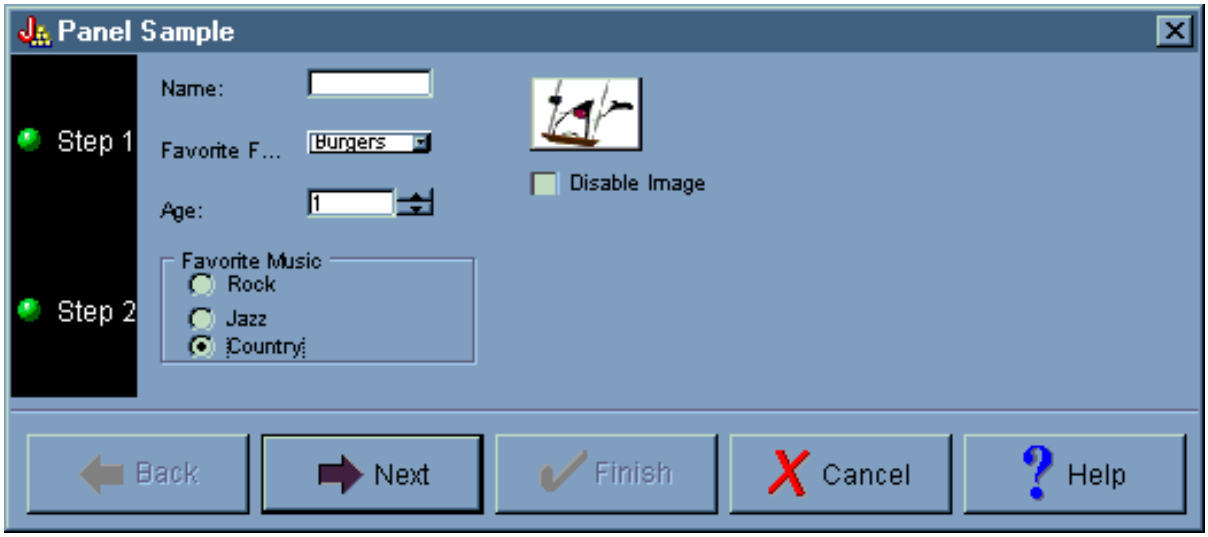


圖 14：遞迴至第一個精靈對話框

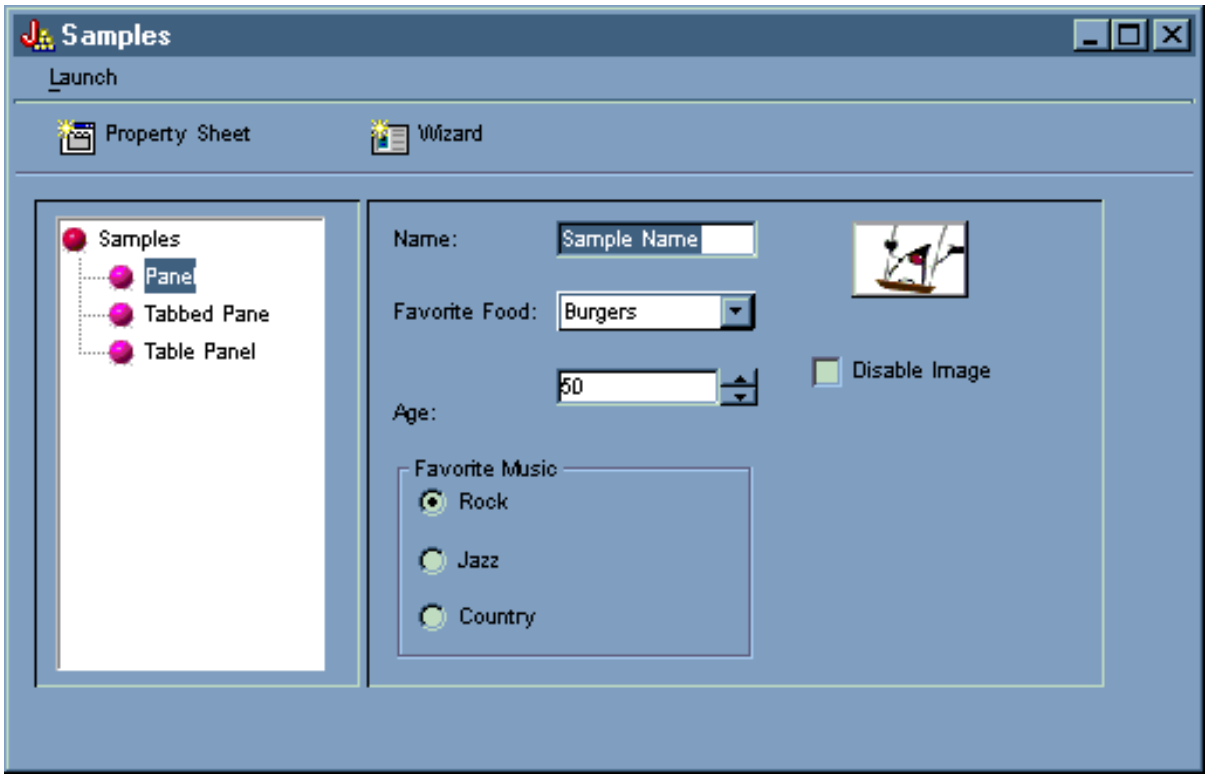


換句話說，程式設計師已經決定使用者不可選取 country 作爲喜歡的音樂形態。

顯示範例

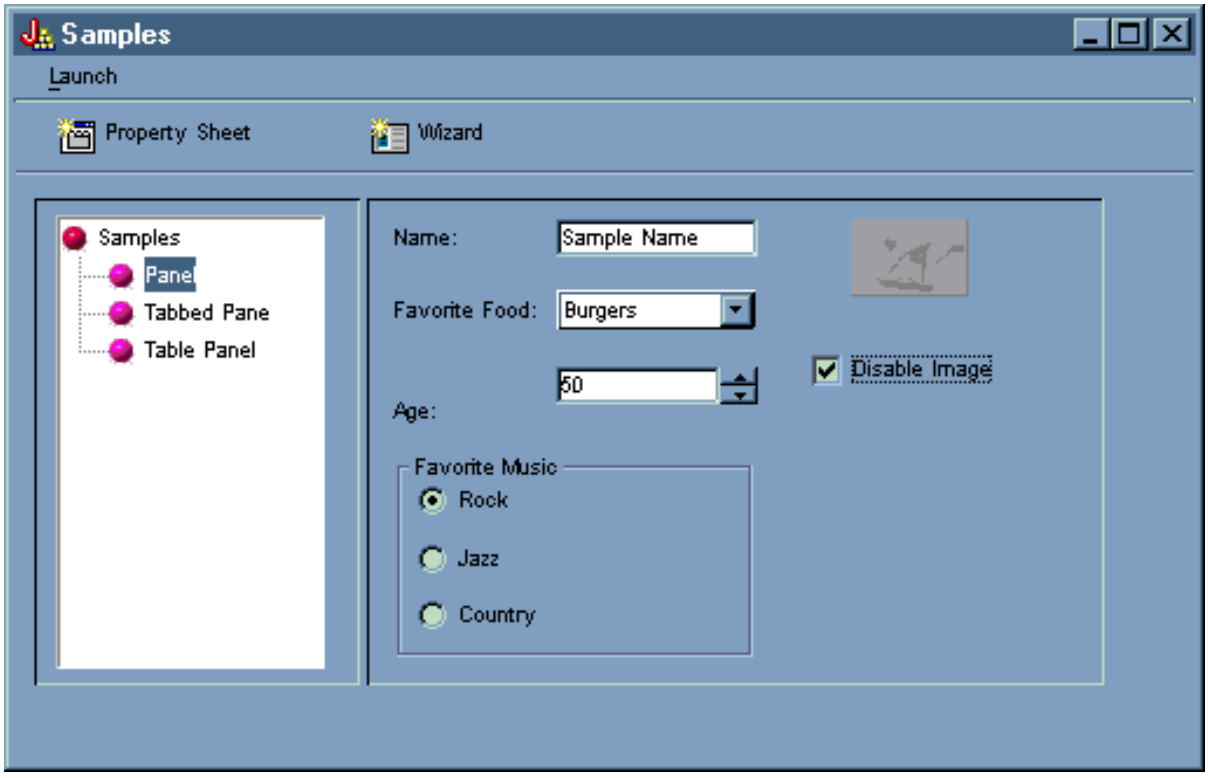
在 GUI Builder 的範例主視窗中，您也可以從工具列下面的左窗格選取其它的功能。圖 15 顯示選取左窗格中的 **Panel** 會在右窗格中顯示 Panel 範例。

圖 15：選取左窗格中的 Panel



此 Panel 範例程式設計爲具有停用影像的選項。選取 **Disable Image**，可以顯示相同螢幕，但其影像會陰影化，如圖 16 所示。

圖 16：選取右窗格中的 Disable Image



此 Panel 範例也說明了下拉清單方塊選項，如圖 17 中所示。

圖 17：在右窗格中，從 Favorite Food 清單中選取一個項目

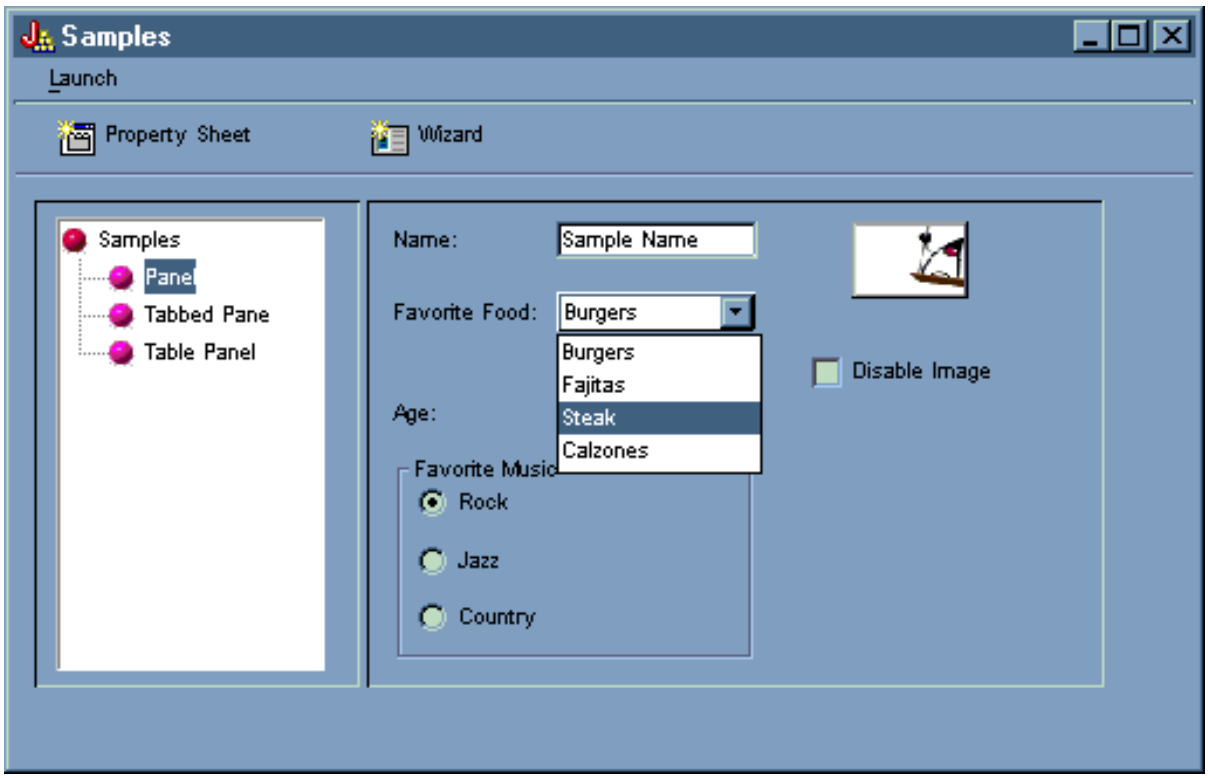


圖 18 顯示在 GUI Builder 範例主視窗的左窗格中選取 **Tabbed Pane**， 會在右窗格中顯示 Tabbed Pane 範例。

圖 18：選取左窗格中的 Tabbed Pane

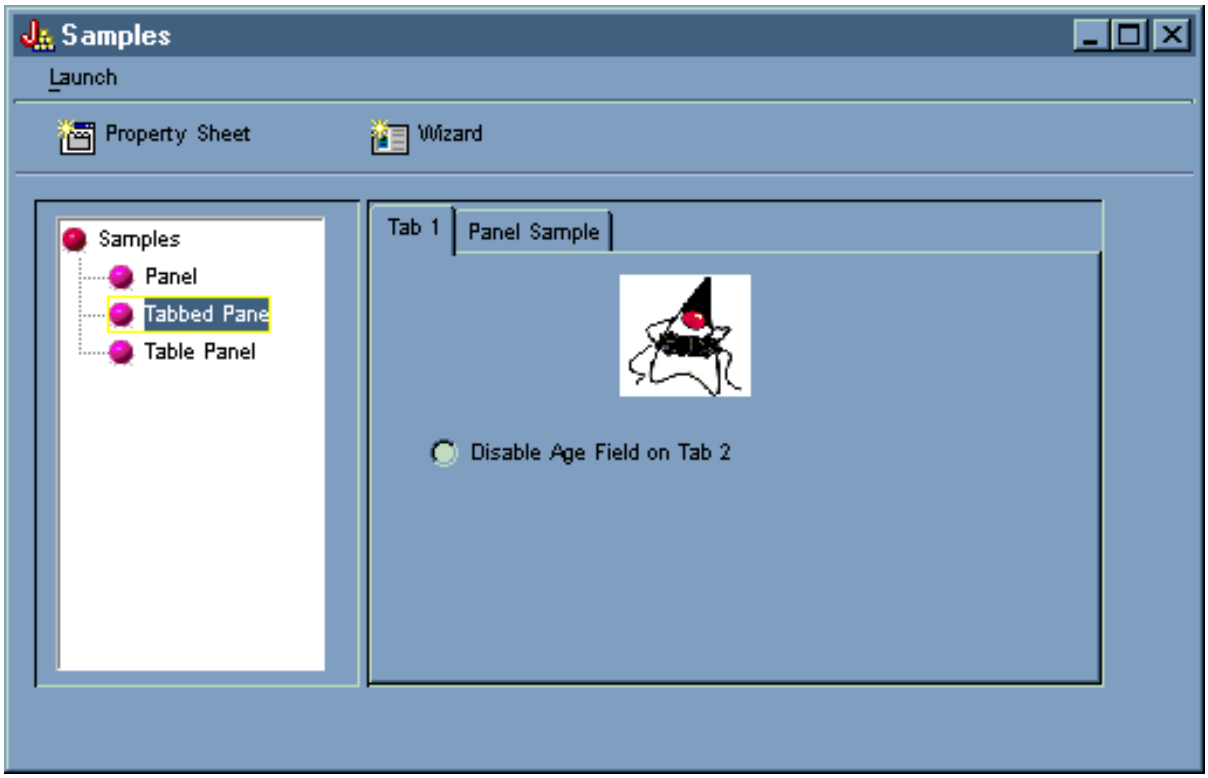
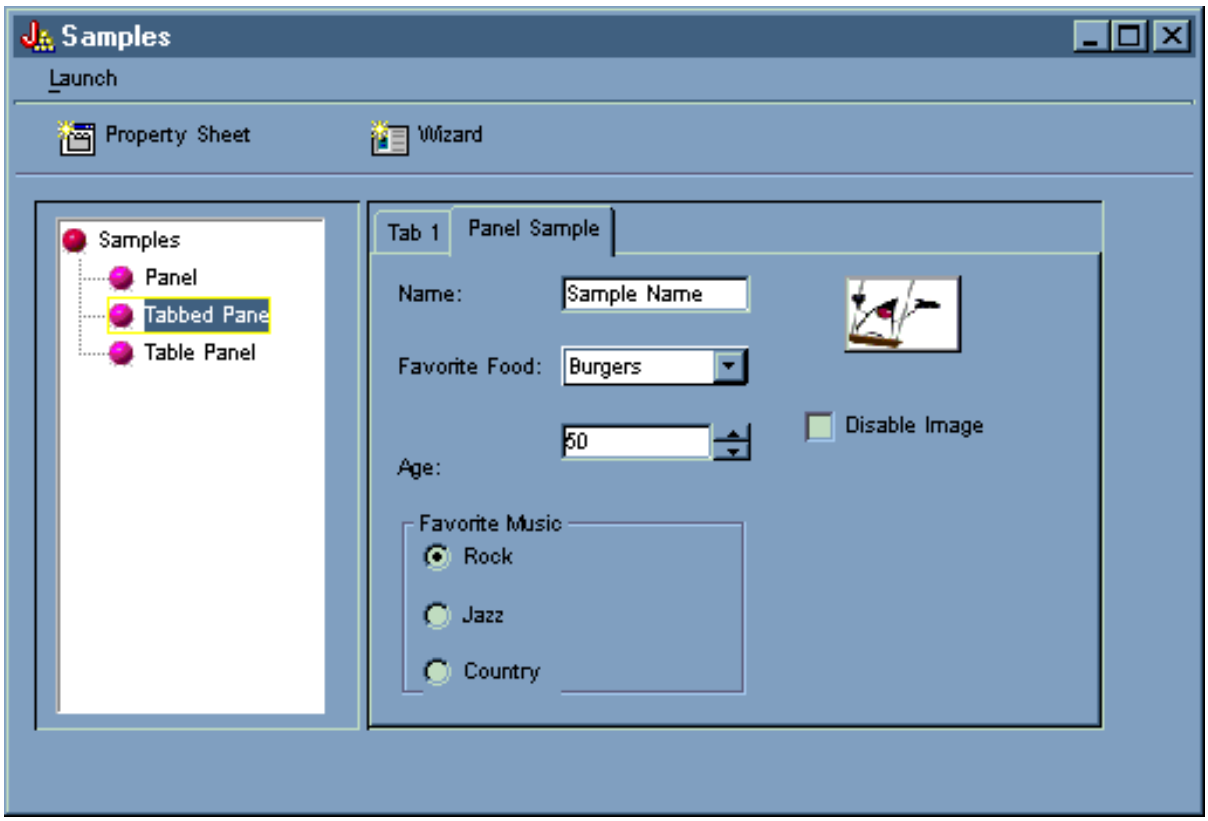


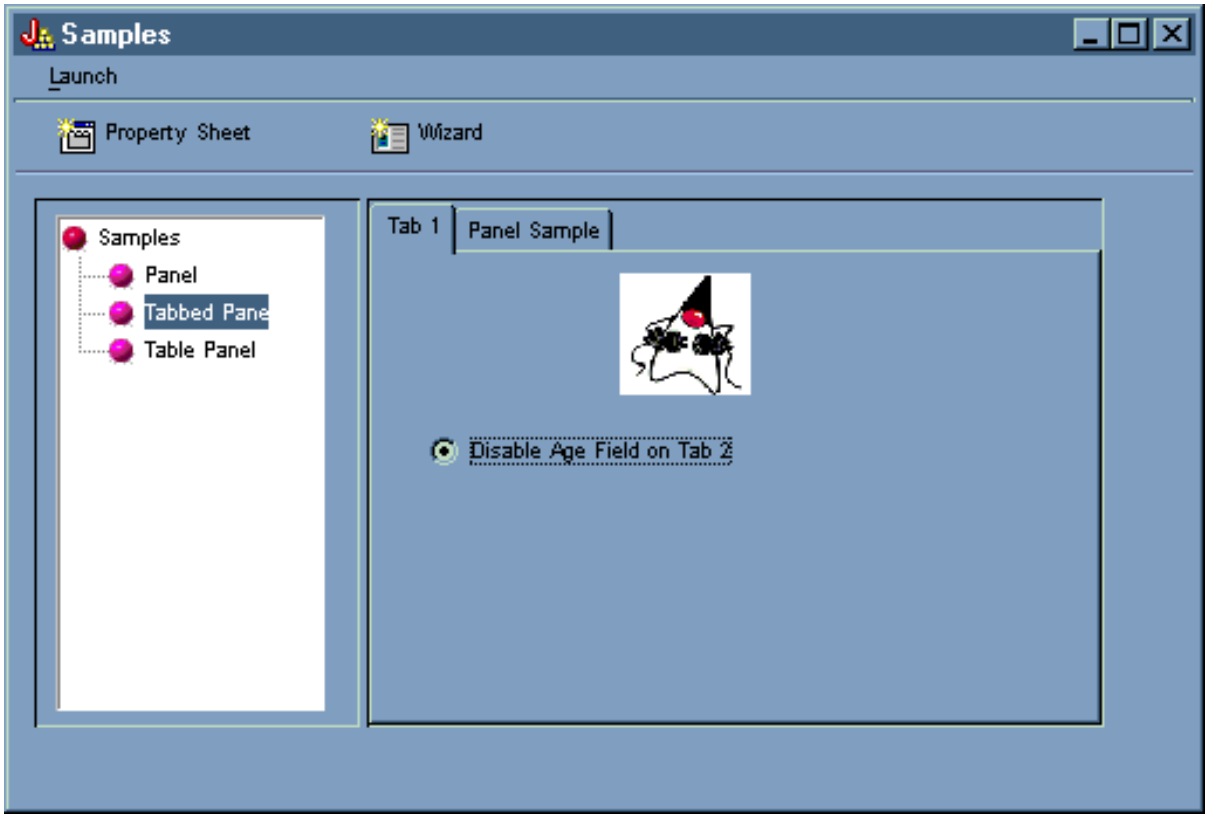
圖 19 顯示選取右窗格中的 **Panel Sample** 標籤的結果。

圖 19：選取右窗格中的 Panel Sample 標籤



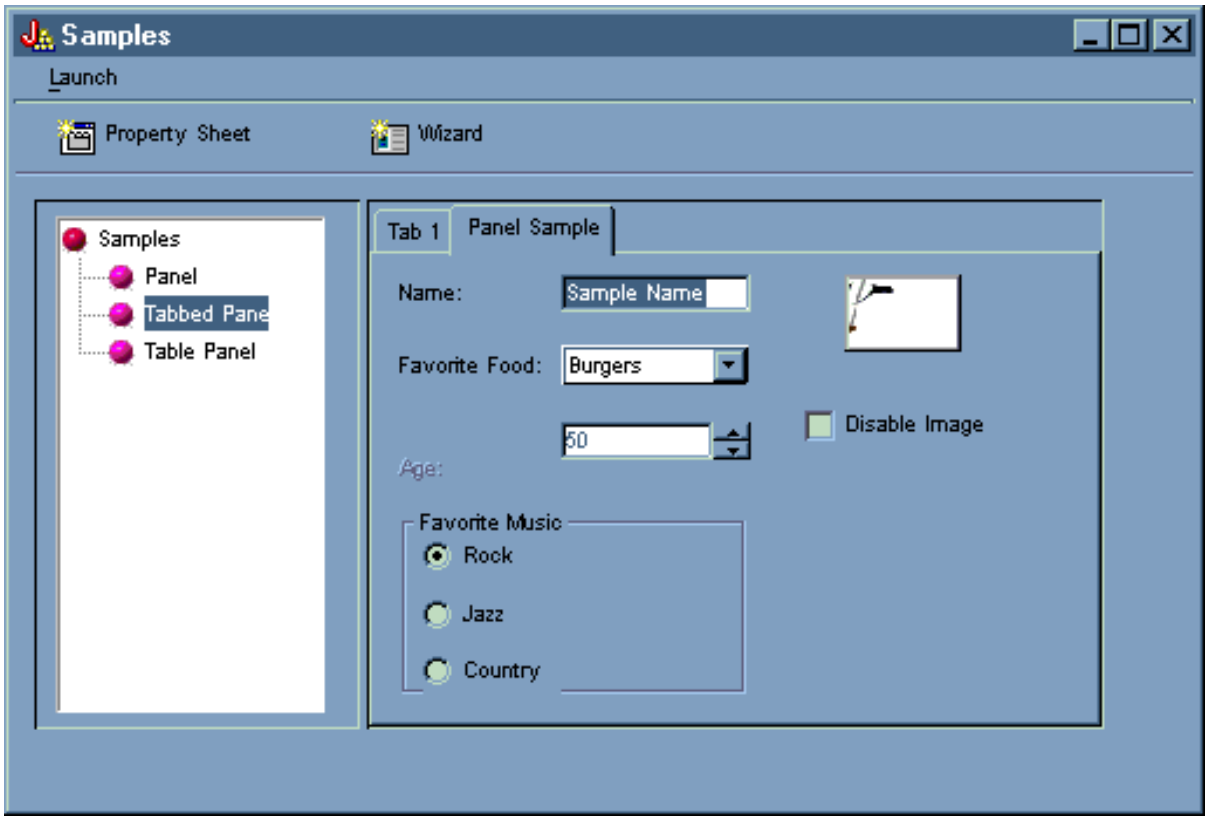
請再次選取 **Tab 1** (右窗格中)，然後按一下 **Disable Age Field on Tab 2**，以取消選取 Tab 1。

圖 20：在右窗格中，選取 Disable Age Field on Tab 2



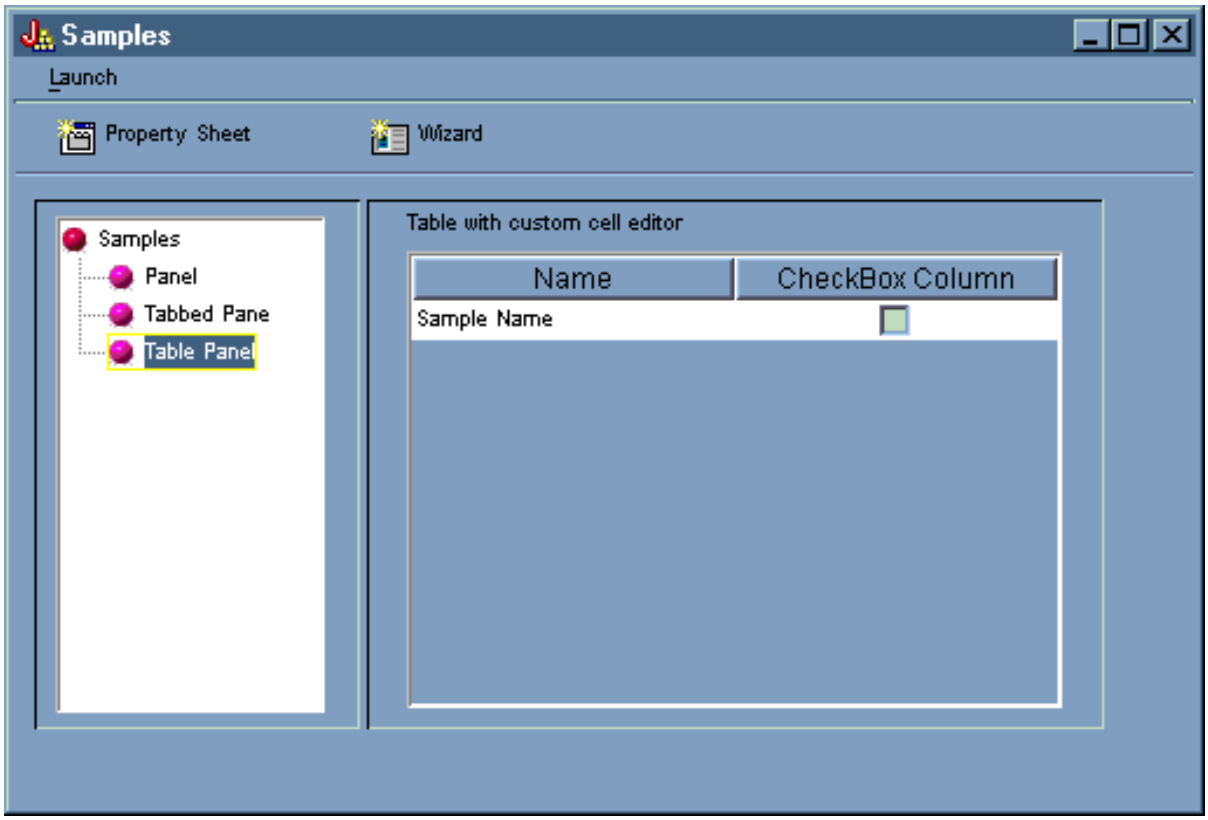
選取 **Disable Age Field on Tab 2** 選項， 會停用 **Panel Sample** 標籤中的 **Age** 欄位，並使該選項的影像變成灰色，如圖 21 中所示。

圖 21：停用 Panel Sample 標籤中 Age 的結果



在 GUI Builder 範例主視窗的左窗格中選取 **Table Pane**，說明使用含有自訂描述器以及自訂資料格編輯器的表格畫面，如圖 22 中所示。

圖 22：選取左窗格中的 Table Panel



HTML 類別範例

下列範例會告訴您使用 HTML 類別的一些方法：

- 範例：使用 BidiOrdering 類別
- 範例：建立 HTMLAlign 物件
- HTMLDocument 類別範例：
  - 範例：使用 HTMLDocument 來建立 HTML 資料
  - 範例：使用 HTMLDocument 來建立 XSL FO 資料
- 範例：使用 HTML 套表類別
- 套表輸入類別範例：
  - 範例：建立 ButtonFormInput 物件
  - 範例：建立 FileFormInput 物件
  - 範例：建立 HiddenFormInput 物件
  - 範例：建立 ImageFormInput 物件
  - 範例：建立 ResetFormInput 物件
  - 範例：建立 SubmitFormInput 物件
  - 範例：建立 TextFormInput 物件
  - 範例：建立 PasswordFormInput 物件
  - 範例：建立 RadioFormInput 物件
  - 範例：建立 CheckboxFormInput 物件
- 範例：建立 HTMLHeading 物件

- 範例：使用 HTMLHyperlink 類別
- 範例：使用 HTMLImage 類別
- HTMLList 範例
  - 範例：建立排序的清單
  - 範例：建立未排序的清單
  - 範例：建立巢狀清單
- 範例：建立 HTMLMeta 標示
- 範例：建立 HTMLParameter 標示
- 範例：建立 HTMLServlet 標示
- 範例：使用 HTMLText 類別
- HTMLTree 範例
  - 範例：使用 HTMLTree 類別
  - 範例：建立一個可遍訪的整合檔案系統樹
- 佈置套表類別：
  - 範例：使用 GridLayoutFormPanel 類別
  - 範例：使用 LineLayoutFormPanel 類別
- 範例：使用 TextAreaFormElement 類別
- 範例：使用 LabelFormOutput 類別
- 範例：使用 SelectFormElement 類別
- 範例：使用 SelectOption 類別
- 範例：使用 RadioFormInputGroup 類別
- 範例：使用 RadioFormInput 類別
- 範例：使用 HTMLTable 類別
  - 範例：使用 HTMLTableCell 類別
  - 範例：使用 HTMLTableRow 類別
  - 範例：使用 HTMLTableHeader 類別
  - 範例：使用 HTMLTableCaption 類別

您也可以一起使用 HTML 及 servlet 類別，就如此範例所示。

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：使用 HTML 套表類別

下面範例將告訴您如何使用 HTML 套表類別。您也可以執行此程式，以檢視範例輸出 。以 ″showHTML″ 方法所使用的 HTML 類別是**粗體**。

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////
//
// This source is an example of using the IBM Toolbox for Java HTML
// package classes, which allow you to easily build HTML Forms.
//
////////////////////////////////////////////////////////////////////////////

package customer;

import java.io.*;
import java.awt.Color;

import javax.servlet.*;
import javax.servlet.http.*;

import com.ibm.as400.access.*;
import com.ibm.as400.util.html.*;


public class HTMLExample extends HttpServlet
{
  // Determines if user already exists in the list of registrants.
  private static boolean found = false;

  // Registration information will be stored here
  String regPath = "c:\\registration.txt";


   public void init(ServletConfig config)
  {

     try
    {
        super.init(config);

    }
    catch (Exception e)
    {
       e.printStackTrace();
    }

  }

  /**
   *  Process the GET request.
   *  @param req The request.
   *  @param res The response.
   **/
      public void doGet (HttpServletRequest req, HttpServletResponse res)
       throws ServletException, IOException
   {
              res.setContentType("text/html");
      ServletOutputStream out = res.getOutputStream();

            // Display the Web using the new HTML classes
            out.println(showHTML());
      out.close();
   }
```

```
public void doPost (HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
{
            String nameStr   = req.getParameter("name");
            String emailStr = req.getParameter("email");
            String errorText= "";

            // Output stream to write to the servlet
    ServletOutputStream out = res.getOutputStream();

            res.setContentType("text/html");

            // Check name & e-mail parameters for valid values
            if (nameStr.length() == 0)
       errorText += "Customer Name not entered.  ";
    if (emailStr.length() == 0)
       errorText += "E-mail not entered.  " ;

    // If name & e-mail have both been provided, continue.
    if (errorText.length() == 0)
    {

       try
       {
          //Create the registration.txt file
          FileWriter f = new FileWriter(regPath, true);
          BufferedWriter output = new BufferedWriter(f);

          //buffered reader for searching the file
          BufferedReader in = new BufferedReader(new FileReader(regPath));

          String line = in.readLine();

          // reset the found flag
          found = false;

          // Check to see if this customer has already registered
          //  or has already used the same e-mail address
          while (!found)
          {
             // if file is empty or end of file reached.
             if (line == null)
                break;

             // if customer already registered
             if ((line.equals("Customer Name: " + nameStr)) ||
                 (line.equals("Email address: " + emailStr)))
             {
                // Output a message to the customer saying they have
                // already registered
                out.println ("<HTML> " +
                             "<TITLE> Toolbox Registration</TITLE> " +
                             "<META HTTP-EQUIV=\"pragma\" content=\"no-cache\"> " +
                             "<BODY BGCOLOR=\"blanchedalmond\" TEXT=\"black\"> " );
                out.println ("<P><HR>" +
                             "<P>" + nameStr +
                             "</B>, you have already registered using that " +
                             "<B>Name</B> or <B>E-mail address</B>." +
                             "<P> Thank You!...<P><HR>");

                // Create a HTMLHyperlink object and display it
                out.println ("<UL><LI>" +
                             new HTMLHyperlink("./customer.HTMLExample",
                                               "Back to Registration Form") +
                             "</UL></BODY></HTML>");
                found = true;
```

```
      break;

   }
   else    // read the next line
      line = in.readLine();

}

// String object to hold data submitted from the HTML Form
String data;

// If the users name or e-mail aren't found in our
// text file, continue.
if (!found)
{
   //-------------------------------------------------------------
   // Insert the new customer info into a file
   output.newLine();
   output.write("Customer Name: " + nameStr);
   output.newLine();
   output.write("Email address: " + emailStr);
   output.newLine();
   //-------------------------------------------------------------


   //-------------------------------------------------------------
   //Getting "USE" checkbox from form
   data = req.getParameter("use");
   if(data != null)
   {
      output.write("Currently Using Toolbox: " + data);
      output.newLine();
   }
   //-------------------------------------------------------------


   //-------------------------------------------------------------
   //Getting "More Information" checkbox from form
   data = req.getParameter("contact");
   if (data != null)
   {
      output.write("Requested More Information: " + data);
      output.newLine();
   }
   //-------------------------------------------------------------


   //-------------------------------------------------------------
   //Getting "AS400 Version" from form
   data = req.getParameter("version");
   if (data != null)
   {
      if (data.equals("multiple versions"))
      {
         data = req.getParameter("MultiList");
         output.write("Multiple Versions: " + data);
      }
      else
         output.write("AS400 Version: " + data);

      output.newLine();
   }
   //-------------------------------------------------------------


   //-------------------------------------------------------------
   //Getting "Current Projects" from form
```

```
data = req.getParameter("interest");
if (data != null)
{
   output.write("Using Java or Interested In: " + data);
   output.newLine();
}
//-------------------------------------------------------------


//-------------------------------------------------------------
//Getting "Platforms" from form
data = req.getParameter("platform");
if (data != null)
{
   output.write("Platforms: " + data);
   output.newLine();
   if (data.indexOf("Other") >= 0)
   {
      output.write("Other Platforms: " + req.getParameter("OtherPlatforms"));
      output.newLine();
   }
}
//-------------------------------------------------------------


//-------------------------------------------------------------
//Getting "Number of iSeries servers" from form
data = req.getParameter("list1");
if (data != null)
{
   output.write("Number of iSeries servers: " + data);
   output.newLine();
}
//-------------------------------------------------------------


//-------------------------------------------------------------
//Getting "Comments" from form
data = req.getParameter("comments");
if (data != null && data.length() > 0)
{
   output.write("Comments: " + data);
   output.newLine();
}
//-------------------------------------------------------------


//-------------------------------------------------------------
//Getting "Attachment"
data = req.getParameter("myAttachment");
if (data != null && data.length() > 0)
{
   output.write("Attachment File: " + data);
   output.newLine();
}
//-------------------------------------------------------------


//-------------------------------------------------------------
//Getting Hidden "Copyright" infomation
data = req.getParameter("copyright");
if (data != null)
{
   output.write(data);
   output.newLine();
}
//-------------------------------------------------------------
```

```
            output.flush();
            output.close();

            // Print a thanks to the customer
            out.println("<HTML>");
            out.println("<TITLE>Thank You!</TITLE>");
            out.println("<META HTTP-EQUIV=\"pragma\" content=\"no-cache\"> ");
            out.println("<BODY BGCOLOR=\"blanchedalmond\">");
            out.println("<HR><P>Thank You for Registering, <B>" + nameStr + "</B>!<P><HR>");

            // Create a HTMLHyperlink object and display it
            out.println("<UL><LI>" +
                        new HTMLHyperlink("./customer.HTMLExample", "Back to Registration Form"));
            out.println("</UL></BODY></HTML>");

         }

       }
      catch (Exception e)
      {
         // Show error in browser
         out.println("<HTML>");
         out.println("<TITLE>ERROR!</TITLE>");
         out.println("<META HTTP-EQUIV=\"pragma\" content=\"no-cache\"> ");
         out.println("<BODY BGCOLOR=\"blanchedalmond\">");
         out.println("<BR><B>Error Message:</B><P>");
         out.println(e + "<P>");

         // Create a HTMLHyperlink object and display it
         out.println("<UL><LI>" +
                     new HTMLHyperlink("./customer.HTMLExample", "Back to Registration Form"));
         out.println("</UL></BODY></HTML>");

         e.printStackTrace();
      }
   }
   else
   {
      // Output a message to the customer saying customer name &
      // e-mail not entered. Please try again
      out.println ("<HTML> " +
                   "<TITLE>Invalid Registration Form</TITLE> " +
                   "<META HTTP-EQUIV=\"pragma\" content=\"no-cache\"> " +
                   "<BODY BGCOLOR=\"blanchedalmond\" TEXT=\"black\"> " );

      out.println ("<HR><B>ERROR</B> in customer data - <P><B>" +
                   errorText +
                   "</B><P>Please Try Again... <HR>");


      // Create a HTMLHyperlink object and display it
      out.println("<UL><LI>" +
                  new HTMLHyperlink("./customer.HTMLExample", "Back to Registration Form") +
                  "</UL></BODY></HTML>");
   }
    // Close the writer
    out.close();


}


public void destroy(ServletConfig config)
{
   // do nothing
```

```
}


public String getServletInfo()
{
    return "My Product Registration";
}

private String showHTML()
{
  // String Buffer to hold HTML Page
  StringBuffer page = new StringBuffer();

  // Create the HTML Form object
  HTMLForm form = new HTMLForm("/servlet/customer.HTMLExample");;
  HTMLText txt;

  // Build the beginning of the HTML Page and add it to the String Buffer
  page.append("<HTML>\n");
  page.append("<TITLE> Welcome!!</TITLE>\n");
  page.append("<HEAD><SCRIPT LANGUAGE=\"JavaScript\">
              function test(){alert(\"This is a sample script executed with a
              ButtonFormInput.\")}</SCRIPT></HEAD>");
  page.append("<META HTTP-EQUIV=\"pragma\" content=\"no-cache\">\n");
  page.append("<BODY BGCOLOR=\"blanchedalmond\" TEXT=\"black\"><BR>\n");

  try
  {
     //-----------------------------------------------------------------
     // Create page title using HTML Text
     txt = new HTMLText("Product Registration");
     txt.setSize(5);
     txt.setBold(true);
     txt.setColor(new Color(199, 21, 133));
     txt.setAlignment(HTMLConstants.CENTER);

     // Add HTML Text to the String Buffer
     page.append(txt.getTag(true) + "<HR><BR>\n");
     //-----------------------------------------------------------------

     //-----------------------------------------------------------------
     // Create a Line Layout
     LineLayoutFormPanel line = new LineLayoutFormPanel();
     txt = new HTMLText("Enter your name and e-mail address:");
     txt.setSize(4);
     line.addElement(txt);

     // Add the Line Layout to String Buffer
     page.append(line.toString());
     page.append("<BR>");
     //-----------------------------------------------------------------

     //-----------------------------------------------------------------
     // Set the HTML Form METHOD
     form.setMethod(HTMLForm.METHOD_POST);
     //-----------------------------------------------------------------

     //-----------------------------------------------------------------
     // Create a Text input for the name.
     TextFormInput user = new TextFormInput("name");
     user.setSize(25);
     user.setMaxLength(40);

     // Create a Text input for the email address.
     TextFormInput email = new TextFormInput("email");
     email.setSize(30);
     email.setMaxLength(40);
```

```
// Create a ImageFormInput
ImageFormInput img =
  new ImageFormInput("Submit Form", "..\\images\\myPiimages/c.gif");
img.setAlignment(HTMLConstants.RIGHT);
//---------------------------------------------------------------------

//---------------------------------------------------------------------
// Create a LineLayoutFormPanel object for the name & e-mail address
LineLayoutFormPanel line2 = new LineLayoutFormPanel();

// Add elements to the line form
line2.addElement(new LabelFormElement("Name:"));
line2.addElement(user);
// Create and add a Label Element to the Line Layout
line2.addElement(new LabelFormElement("E-mail:"));
line2.addElement(email);
line2.addElement(img);
//---------------------------------------------------------------------

//---------------------------------------------------------------------
// Create Questions line layout
LineLayoutFormPanel line3 = new LineLayoutFormPanel();

// Add elements to the line layout
line3.addElement(new LineLayoutFormPanel());
line3.addElement(new
  CheckboxFormInput("use",
                    "yes",
                    "Do you currently use the Toolbox?",
                    false));
line3.addElement(new LineLayoutFormPanel());
line3.addElement(new CheckboxFormInput(
   "contact",
   "yes",
   "Would you like information on future Toolbox releases?",
   true));
line3.addElement(new LineLayoutFormPanel());
//---------------------------------------------------------------------

//---------------------------------------------------------------------
// Create Version Radio Group
RadioFormInputGroup group = new RadioFormInputGroup("version");

// Add Radio Form Inputs to the Group
group.add(new RadioFormInput("version", "v3r2", "V3R2", false));
group.add(new RadioFormInput("version", "v4r1", "V4R1", false));
group.add(new RadioFormInput("version", "v4r2", "V4R2", false));
group.add(new RadioFormInput("version", "v4r3", "V4R3", false));
group.add(new RadioFormInput("version", "v4r4", "V4R4", false));
group.add(new
  RadioFormInput("version",
                 "multiple versions",
                 "Multiple Versions? Which ones:",
                 false));

//Create a Select Form Element
SelectFormElement mlist = new SelectFormElement("MultiList");
mlist.setMultiple(true);
mlist.setSize(3);

//Create the Options for the Select Form Element
SelectOption option1 = mlist.addOption("V3R2", "v3r2");
SelectOption option2 = mlist.addOption("V4R1", "v4r1");
SelectOption option3 = mlist.addOption("V4R2", "v4r2");
SelectOption option4 = mlist.addOption("V4R3", "v4r3");
SelectOption option5 = mlist.addOption("V4R4", "v4r4");
```

```
// Create HTML text
txt = new HTMLText("Current Server Level:");
txt.setSize(4);

// Create Grid Layout
GridLayoutFormPanel grid1 = new GridLayoutFormPanel(3);

// Add radio group & select form element to the grid
grid1.addElement(txt);
grid1.addElement(group);
grid1.addElement(mlist);
//---------------------------------------------------------------------

//---------------------------------------------------------------------
// Create Grid Layout for interests
GridLayoutFormPanel grid2 = new GridLayoutFormPanel(1);
txt = new HTMLText("Current Projects or Area of Interest: " +
                   "(check all that apply)");
txt.setSize(4);

// Add elements to Grid Layout
grid2.addElement(new LineLayoutFormPanel());
grid2.addElement(txt);
// Create and add a Checkbox to the Grid Layout
grid2.addElement(new
  CheckboxFormInput("interest", "applications", "Applications", true));
grid2.addElement(new
  CheckboxFormInput("interest", "applets", "Applets", false));
grid2.addElement(new
  CheckboxFormInput("interest", "servlets", "Servlets", false));
//---------------------------------------------------------------------

//---------------------------------------------------------------------
// Create Line Layout for platforms
LineLayoutFormPanel line4 = new LineLayoutFormPanel();
txt = new HTMLText("Client Platforms Used: " +
                   "(check all that apply)");
txt.setSize(4);

// Add elements to Line Layout
line4.addElement(new LineLayoutFormPanel());
line4.addElement(txt);
line4.addElement(new LineLayoutFormPanel());
line4.addElement(new CheckboxFormInput("platform",
                                       "95",
                                       "Windows95",
                                       false));
line4.addElement(new CheckboxFormInput("platform",
                                       "98",
                                       "Windows98",
                                       false));
line4.addElement(new CheckboxFormInput("platform",
                                       "NT",
                                       "WindowsNT",
                                       false));
line4.addElement(new CheckboxFormInput("platform",
                                       "OS2",
                                       "OS/2",
                                       false));
line4.addElement(new CheckboxFormInput("platform",
                                       "AIX",
                                       "AIX",
                                       false));
line4.addElement(new CheckboxFormInput("platform",
                                       "Linux",
                                       "Linux",
```

```
                                        false));
line4.addElement(new CheckboxFormInput("platform",
                                        "AS400",
                                        "iSeries",
                                        false));
line4.addElement(new CheckboxFormInput("platform",
                                        "Other",
                                        "Other:",
                                        false));

TextFormInput other = new TextFormInput("OtherPlatforms");
other.setSize(20);
other.setMaxLength(50);

line4.addElement(other);
//-----------------------------------------------------------------------

//-----------------------------------------------------------------------
// Create a Line Layout for number of servers
LineLayoutFormPanel grid3 = new LineLayoutFormPanel();

txt = new HTMLText(
   "How many iSeries servers do you have?");
txt.setSize(4);

// Create a Select Form Element for number of servers owned
SelectFormElement list = new SelectFormElement("list1");
// Create and add the Select Options to the Select Form Element List
SelectOption opt0 = list.addOption("0", "zero");
SelectOption opt1 = list.addOption("1", "one", true);
SelectOption opt2 = list.addOption("2", "two");
SelectOption opt3 = list.addOption("3", "three");
SelectOption opt4 = list.addOption("4", "four");
SelectOption opt5 = new SelectOption("5+", "FiveOrMore", false);
list.addOption(opt5);

// Add Elements to the Grid Layout
grid3.addElement(new LineLayoutFormPanel());
grid3.addElement(txt);
grid3.addElement(list);
//-----------------------------------------------------------------------

//-----------------------------------------------------------------------
// Create a Grid Layout for Product Comments
GridLayoutFormPanel grid4 = new GridLayoutFormPanel(1);
txt = new HTMLText("Product Comments:");
txt.setSize(4);

// Add elements to the Grid Layout
grid4.addElement(new LineLayoutFormPanel());
grid4.addElement(txt);
// Create a Text Area Form
grid4.addElement(new TextAreaFormElement("comments", 5, 75));
grid4.addElement(new LineLayoutFormPanel());
//-----------------------------------------------------------------------

//-----------------------------------------------------------------------
// Create a Grid Layout
GridLayoutFormPanel grid5 = new GridLayoutFormPanel(2);
txt = new HTMLText("Would you like to sign on to a server?");
txt.setSize(4);

// Create a Text input and Label for the system name.
TextFormInput sys = new TextFormInput("system");
LabelFormElement sysLabel = new LabelFormElement("System:");

// Create a Text input and Label for the userid.
```

```
   TextFormInput uid = new TextFormInput("uid");
   LabelFormElement uidLabel = new LabelFormElement("UserID");

   // Create a Password input and Label for the password.
   PasswordFormInput pwd = new PasswordFormInput("pwd");
   LabelFormElement pwdLabel = new LabelFormElement("Password");

   // Add the Text inputs, password inputs, and Labels to the grid
   grid5.addElement(sysLabel);
   grid5.addElement(sys);
   grid5.addElement(uidLabel);
   grid5.addElement(uid);
   grid5.addElement(pwdLabel);
   grid5.addElement(pwd);
   //-----------------------------------------------------------------------

   //-----------------------------------------------------------------------
   // Add the various panels created to the HTML Form
   // in the order you wish them to appear
   form.addElement(line2);
   form.addElement(line3);
   form.addElement(grid1);
   form.addElement(grid2);
   form.addElement(line4);
   form.addElement(grid3);
   form.addElement(grid4);
   form.addElement(txt);
   form.addElement(new LineLayoutFormPanel());
   form.addElement(grid5);
   form.addElement(new LineLayoutFormPanel());
   form.addElement(
      new HTMLText("Submit an attachment Here: <br />"));
   // Add a File Input to the form
   form.addElement(new FileFormInput("myAttachment"));
   form.addElement(new ButtonFormInput("button",
                                       "TRY ME!",
                                       "test()"));
   // Adds a empty Line Layout, which in turn
   // adds a line break <br /> to the form
   form.addElement(new LineLayoutFormPanel());
   form.addElement(new LineLayoutFormPanel());
   form.addElement(new SubmitFormInput("submit", "Register"));
   form.addElement(new LineLayoutFormPanel());
   form.addElement(new LineLayoutFormPanel());
   form.addElement(new ResetFormInput("reset", "Reset"));
   // Add a Hidden Input to the form
   form.addElement(new
     HiddenFormInput("copyright",
                     "(C) Copyright IBM Corp. 1999, 1999"));
   //-----------------------------------------------------------------------

   // Add the entire HTML Form to the String Buffer
   page.append(form.toString());

}
catch(Exception e)
{
   e.printStackTrace();
}

// Add the Ending HTML tags to the Buffer
page.append("</BODY>\n");
page.append("</HTML>\n");

// Return the entire HTML page string
```

```
      return page.toString();
   }
}
```

HTML 類別範例輸出

執行 HTML 類別範例，您可以取得一些可能的範例輸出：

- Customer Name: Fred Flinstone
  Email address: flinstone@bedrock.com
  Currently Using Toolbox: yes
  Requested More Information: yes
  Multiple Versions: v4r2,v4r4
  Using Java or Interested In: applications,servlets
  Platforms: NT,Linux
  Number of iSeries servers: three
  Comments: The Toolbox is being used by our entire Programming department to build customer applications!
  Attachment File: U:\wiedrich\servlet\temp.html
  (C) Copyright IBM Corp. 1999, 1999
- Customer Name: Barney Rubble
  Email address: rubble@bedrock.com
  Currently Using Toolbox: yes
  AS400 Version: v4r4
  Using Java or Interested In: servlets
  Platforms: OS2
  Number of iSeries servers: FiveOrMore
  (C) Copyright IBM Corp. 1999, 1999
- Customer Name: George Jetson
  Email address: jetson@sprocket.com
  Requested More Information: yes
  AS400 Version: v4r2
  Using Java or Interested In: applications
  Platforms: NT,Other
  Other Platforms: Solaris
  Number of iSeries servers: one
  Comments: This is my first time using this! Very Cool!
  (C) Copyright IBM Corp. 1999, 1999
- Customer Name: Clark Kent
  Email address: superman@krypton.com
  AS400 Version: v4r2
  Number of iSeries servers: one
  (C) Copyright IBM Corp. 1999, 1999

範例：使用 HTMLTree 類別

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
///////////////////////////////////////////////////////////////////////////
//
// This source is an example of using the IBM Toolbox for Java HTML
// package classes, which allow you to easily build HTML and File Trees.
//
///////////////////////////////////////////////////////////////////////////

import java.io.File;
import java.io.PrintWriter;
import java.io.IOException;

import java.util.Vector;
import java.util.Properties;

import javax.servlet.*;
import javax.servlet.http.*;

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.Trace;
import com.ibm.as400.access.IFSJavaFile;
import com.ibm.as400.util.html.HTMLMeta;
import com.ibm.as400.util.html.HTMLTree;
import com.ibm.as400.util.html.HTMLTreeElement;
import com.ibm.as400.util.html.URLParser;
import com.ibm.as400.util.html.DirFilter;
import com.ibm.as400.util.html.FileTreeElement;
import com.ibm.as400.util.servlet.ServletHyperlink;

/**
  *  An example of using the HTMLTree and FileTreeElement classes in a servlet.
 **/
public class TreeNav extends HttpServlet
{
    public void init(ServletConfig config)
      throws ServletException
   {
       super.init(config);

      // The Toolbox uses a set of default icons to represents expanded,
      // collapsed, and documents within the HTMLTree. To enhance those icons,
      // the Toolbox ships three gifs (expanded.gif, collapsed.gif, bullet.gif)
      // in the jt400Servlet.jar file.  Browsers do not have the ability to
      // find gifs in a jar or zip file, so those images need to be extracted
      // from the jar file and placed in the appropriate webserver directory
      // (by default it is the /html directory). Then uncomment the following
      // lines of code and specify the correct location in the these set
      // methods. The location can be absolute or relative.

      HTMLTreeElement.setExpandedGif("/images/expanded.gif");
      HTMLTreeElement.setCollapsedGif("/images/collapsed.gif");
      HTMLTreeElement.setDocGif("/images/bullet.gif");
   }

   /**
    *  Process the GET request.
    *  @param req The request.
    *  @param res The response.
    **/
      public void doGet (HttpServletRequest req, HttpServletResponse resp)
       throws ServletException, IOException
   {
      HttpSession session = req.getSession(true);
      HTMLTree fileTree = (HTMLTree)session.getValue("filetree");

      // If this session does not already have a file tree, then
      // create the initial tree.
      if (fileTree == null)
```

```
        fileTree = createTree(req, resp, req.getRequestURI());

    // Set the Http servlet request on the HTMLTree.
    fileTree.setHttpServletRequest(req);

    resp.setContentType("text/html");

    PrintWriter out = resp.getWriter();
    out.println("<html>\n");
    out.println(new HTMLMeta("Expires","Mon, 03 Jan 1990 13:00:00 GMT"));
    out.println("<body>\n");

    // Get the tag for the HTMLTree.
    out.println(fileTree.getTag());

    out.println("</body>\n");
    out.println("</html>\n");
    out.close();

    // Set the session tree value, so when entering this servlet for
    // the second time, the FileTree object will be reused.
    session.putValue("filetree", fileTree);
}

/**
 *  Process the POST request.
 *  @param req The request.
 *  @param res The response.
 **/
public void doPost (HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
{
            res.setContentType("text/html");
    ServletOutputStream out = res.getOutputStream();
}

/**
 *  This method will create the initial HTMLTree.
 **/
    private HTMLTree createTree(HttpServletRequest req, HttpServletResponse resp, String uri)
{
    // Create an HTMLTree object.
    HTMLTree tree = new HTMLTree(req);

     try
    {
                // Create a URLParser object.
       URLParser urlParser = new URLParser(uri);

       AS400 sys = new AS400(CPUStatus.systemName_, "javactl", "jteam1");

       // Create a File object and set the root IFS directory.
       IFSJavaFile root = new IFSJavaFile(sys, "/QIBM");

                // Create a Filter and list all of the directories.
       DirFilter filter = new DirFilter();
                //File[] dirList = root.listFiles(filter);

       // Get the list of files that satisfy the directory filter.
       String[] list = root.list(filter);

       File[] dirList = new File[list.length];

       // We don't want to require webservers to use JDK1.2 because
                // most webserver JVM's are slower to upgrade to the latest JDK level.
                // The most efficient way to create these file objects is to use
                // the listFiles(filter) method in JDK1.2 which would be done
```

```
                // like the following, instead of using the list(filter) method
                // and then converting the returned string arrary into the appropriate
                // File array.
                // File[] dirList = root.listFiles(filter);

         for (int j=0; j<dirList.length; ++j)
         {
            if (root instanceof IFSJavaFile)
               dirList[j] = new IFSJavaFile((IFSJavaFile)root, list[j]);
            else
               dirList[j] = new File(list[j]);
         }

         for (int i=0; i<dirList.length; i++)
         {
            // Create a FileTreeElement for each directory in the list.
            FileTreeElement node = new FileTreeElement(dirList[i]);

            // Create a ServletHyperlink for the expand/collapse icons.
            ServletHyperlink sl = new ServletHyperlink(urlParser.getURI());
            //sl.setHttpServletResponse(resp);
            node.setIconUrl(sl);

            // Create a ServletHyperlink to the TreeList servlet, which will
            // display the contents of thie FileTreeElement (directory).
            ServletHyperlink tl = new ServletHyperlink("/servlet/TreeList");
            tl.setTarget("list");

            // If the ServletHyperlink doesn't have a name, then set it to the
            // name of the directory.
            if (tl.getText() == null)
               tl.setText(dirList[i].getName());

            // Set the TextUrl for the FileTreeElement.
            node.setTextUrl(tl);

            // Add the FileTreeElement to the HTMLTree.
            tree.addElement(node);
         }
      }
      catch (Exception e)
      {
        e.printStackTrace();
      }

      return tree;
   }

   public void destroy(ServletConfig config)
   {
      // do nothing
   }

   public String getServletInfo()
   {
       return "FileTree Navigation";
   }
}
```

範例：建立可遍訪的整合檔案系統樹 (檔案 3 之 1)

此範例程式碼結合了其他兩個範例檔中的程式碼，可顯示 servlet 中的 HTMLTree 和 FileListElement。範例中的三個檔案爲：

- FileTreeExample.java - 此檔案會產生 HTML 訊框並啓動 servlet

- TreeNav.java - 建置及管理樹
- TreeList.java - 顯示在 TreeNav.java 類別中所做的選擇內容

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////
//
// This source is an example of using the IBM Toolbox for Java HTML package
// classes, which allow you to easily build HTML and File Trees.
//
////////////////////////////////////////////////////////////////////////////

import java.io.PrintWriter;
import java.io.IOException;

import javax.servlet.*;
import javax.servlet.http.*;

import com.ibm.as400.util.html.HTMLMeta;


//
// An example of using frames to display an HTMLTree and FileListElement
// in a servlet.
//

public class FileTreeExample extends HttpServlet
{
    public void init(ServletConfig config)
      throws ServletException
   {
       super.init(config);
   }

   /**
    *  Process the GET request.
    *  @param req The request.
    *  @param res The response.
    **/

      public void doGet (HttpServletRequest req, HttpServletResponse resp)
       throws ServletException, IOException
   {
      resp.setContentType("text/html");

      // Set up two frames. The first, a navigation frame, will display
// the HTMLTree, which will contain FileTreeElements and allow
      // navigation of the File system.The second frame will display/list
// the contents of a selected directory from the navigation frame.
      PrintWriter out = resp.getWriter();
      out.println("<html>\n");
      out.println(new HTMLMeta("Expires","Mon, 04 Jan 1990 13:00:00 GMT"));
      out.println("<frameset cols=\"25%,*\">");
      out.println("<frame frameborder=\"5\" src=\"/servlet/TreeNav\" name=\"nav\">");
      out.println("<frame frameborder=\"3\" src=\"/servlet/TreeList\" name=\"list\">");
      out.println("</frameset>");
      out.println("</html>\n");
      out.close();
   }

   /**
    *  Process the POST request.
    *  @param req The request.
    *  @param res The response.
    **/
```

```
   public void doPost (HttpServletRequest req, HttpServletResponse res)
       throws ServletException, IOException
   {
              res.setContentType("text/html");
      ServletOutputStream out = res.getOutputStream();
   }

   public void destroy(ServletConfig config)
   {
       // do nothing
   }

    public String getServletInfo()
   {
      return "FileTree Servlet";
   }
}
```

範例：建立可遍訪的整合檔案系統樹 (檔案 3 之 2)

此範例程式碼與其它兩個範例檔中的程式碼相連結，可在 servlet 中顯示 HTMLTree 與 FileListElement。範例中的三個檔案如下：

- FileTreeExample.java - 產生 HTML 框架並啓動 servlet
- TreeNav.java - 此檔案可建置及管理樹狀結構
- TreeList.java - 顯示在 TreeNav.java 類別中所選擇的內容

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////////
//
// This source is an example of using the IBM Toolbox for Java HTML
// package classes, which allow you to easily build HTML and File Trees.
//
///////////////////////////////////////////////////////////////////////////////

import java.io.File;
import java.io.PrintWriter;
import java.io.IOException;

import javax.servlet.*;
import javax.servlet.http.*;

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.IFSJavaFile;
import com.ibm.as400.util.html.HTMLMeta;
import com.ibm.as400.util.html.HTMLTree;
import com.ibm.as400.util.html.HTMLTreeElement;
import com.ibm.as400.util.html.URLParser;
import com.ibm.as400.util.html.DirFilter;
import com.ibm.as400.util.html.FileTreeElement;
import com.ibm.as400.util.servlet.ServletHyperlink;

//
// An example of using the HTMLTree and FileTreeElement classes
// in a servlet.
//

public class TreeNav extends HttpServlet
{
    private AS400 sys_;

    public void init(ServletConfig config)
      throws ServletException
   {
```

```
    super.init(config);

    // Create an AS400 object.
   sys_ = new AS400("mySystem", "myUserID", "myPassword");

   // IBM Toolbox for Java uses a set of default icons to represents expanded,
   // collapsed, and documents within the HTMLTree. To enhance those icons,
   // IBM Toolbox for Java ships three gifs (expanded.gif, collapsed.gif, bullet.gif)
   // in the jt400Servlet.jar file.  Browsers do not have the ability to find
   // gifs in a jar or zip file, so you need to extract those images from the
   // jar file and place them in the appropriate webserver directory (by default
   // it is the /html directory). Then change the following lines of code to
   // specify the correct location in the set methods.  The location can be
   // absolute or relative.

   HTMLTreeElement.setExpandedGif("http://myServer/expanded.gif");
   HTMLTreeElement.setCollapsedGif("http://myServer/collapsed.gif");
   HTMLTreeElement.setDocGif("http://myServer/bullet.gif");
}

/**
 *  Process the GET request.
 *  @param req The request.
 *  @param res The response.
 **/

   public void doGet (HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException
{
   // Use session data to remember the state of the tree.
   HttpSession session = req.getSession(true);
   HTMLTree fileTree = (HTMLTree)session.getValue("filetree");

   // If this session does not already have a file tree, then
   // create the initial tree.
   if (fileTree == null)
      fileTree = createTree(req, resp, req.getRequestURI());

   // Set the Http servlet request on the HTMLTree.
   fileTree.setHttpServletRequest(req);

   resp.setContentType("text/html");

   PrintWriter out = resp.getWriter();
   out.println("<html>\n");
   out.println(new HTMLMeta("Expires","Mon, 03 Jan 1990 13:00:00 GMT"));
   out.println("<body>\n");

   // Get the tag for the HTMLTree.
   out.println(fileTree.getTag());

   out.println("</body>\n");
   out.println("</html>\n");
   out.close();

   // Set the session tree value, so when entering this servlet for
   // the second time, the FileTree object will be reused.
   session.putValue("filetree", fileTree);
}

/**
 *  Process the POST request.
 *  @param req The request.
 *  @param res The response.
 **/

public void doPost (HttpServletRequest req, HttpServletResponse res)
```

```
      throws ServletException, IOException
   {
      res.setContentType("text/html");
      ServletOutputStream out = res.getOutputStream();
   }

   /**
    *  This method will create the initial HTMLTree.
    **/

   private HTMLTree createTree(HttpServletRequest req,
                               HttpServletResponse resp, String uri)
   {
      // Create an HTMLTree object.
      HTMLTree tree = new HTMLTree(req);

        try
        {
          // Create a URLParser object.
          URLParser urlParser = new URLParser(uri);

          // Create a File object and set the root IFS directory.
          IFSJavaFile root = new IFSJavaFile(sys_, "/QIBM");

          // Create a Filter.
          DirFilter filter = new DirFilter();

          // Get the list of files that satisfy the directory filter.
          String[] list = root.list(filter);

          File[] dirList = new File[list.length];

          // We don't want to require webservers to use JDK1.2 because
          // most webserver JVM's are slower to upgrade to the latest
          // JDK level. The most efficient way to create these file objects
// is to use the listFiles(filter) method in JDK1.2 which would
          // be done like the following, instead of using the list(filter)
          // method and then converting the returned string arrary into the
          // appropriate File array.
          // File[] dirList = root.listFiles(filter);

          for (int j=0; j<dirList.length; ++j)
          {
            if (root instanceof IFSJavaFile)
               dirList[j] = new IFSJavaFile((IFSJavaFile)root, list[j]);
            else
               dirList[j] = new File(list[j]);
          }

          for (int i=0; i<dirList.length; i++)
          {
            // Create a FileTreeElement for each directory in the list.
            FileTreeElement node = new FileTreeElement(dirList[i]);

            // Create a ServletHyperlink for the expand/collapse icons.
            ServletHyperlink sl = new ServletHyperlink(urlParser.getURI());
            sl.setHttpServletResponse(resp);
            node.setIconUrl(sl);

            // Create a ServletHyperlink to the TreeList servlet, which will
            // display the contents of thie FileTreeElement (directory).
            ServletHyperlink tl = new ServletHyperlink("/servlet/TreeList");
            tl.setTarget("list");

            // If the ServletHyperlink doesn't have a name, then set it to the
            // name of the directory.
            if (tl.getText() == null)
```

```
              tl.setText(dirList[i].getName());

            // Set the TextUrl for the FileTreeElement.
            node.setTextUrl(tl);

            // Add the FileTreeElement to the HTMLTree.
            tree.addElement(node);
          }

          sys_.disconnectAllServices();
        }

        catch (Exception e)
        {
          e.printStackTrace();
        }

        return tree;
    }

    public void destroy(ServletConfig config)
    {
       // do nothing
    }

    public String getServletInfo()
    {
        return "FileTree Navigation";
    }
}
```

範例：建立可遍訪的整合檔案系統樹 (檔案 3 之 3)

此範例程式碼與其它兩個範例檔中的程式碼相連結，可在 servlet 中顯示 HTMLTree 與 FileListElement。範例中的三個檔案如下：

- FileTreeExample.java - 產生 HTML 框架並啓動 servlet
- TreeNav.java - 建置及管理樹狀結構
- TreeList.java - 本檔案顯示在 TreeNav.java 類別中產生的選項內容

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////////
//
// This source is an example of using the IBM Toolbox for Java HTML
// package classes, which allow you to easily build HTML and File Lists.
//
///////////////////////////////////////////////////////////////////////////////

import java.io.PrintWriter;
import java.io.IOException;
import java.io.File;

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.Trace;
import com.ibm.as400.access.IFSJavaFile;
import com.ibm.as400.util.html.HTMLMeta;
import com.ibm.as400.util.html.HTMLHeading;
import com.ibm.as400.util.html.HTMLConstants;
import com.ibm.as400.util.html.FileListElement;
import com.ibm.as400.util.html.*;

import javax.servlet.*;
import javax.servlet.http.*;
```

```
/**
  *An example of using the FileListElement class in a servlet.
 **/
public class TreeList extends HttpServlet
{

     private AS400 sys_;

    /**
     *  Process the GET request.
     *  @param req The request.
     *  @param res The response.
     **/
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
     throws ServletException, IOException
    {
        resp.setContentType("text/html");

         try
        {

            PrintWriter out = resp.getWriter();
            out.println("<html>\n");
            out.println(new HTMLMeta("Expires",
                                     "Mon, 02 Jan 1990 13:00:00 GMT"));
            out.println("<body>\n");

            // If the path parameter is not null, then the user has selected an
            // element from the FileTreeElement list in the navigation frame.
            if (req.getPathInfo() != null)
            {
                // Create a FileListElement passing in an AS400 system object and
                // the Http servlet request. The request will contain the necessary
                // path information to list out the contents of the FileTreeElement
                // (directory) selected.
                FileListElement fileList = new FileListElement(sys_, req);

                //  Alternately, create a FileListElement from a NetServer share name
                //  and share path.
                //  FileListElement fileList =
                      new FileListElement(sys_, req, "TreeShare",
                                          "/QIBM/ProdData/HTTP/Public/jt400");


                // Display the FileListElement contents.
                out.println(fileList.list());
            }
            // Display this HTMLHeading if no FileTreeElement has been selected.
            else
            {
                HTMLHeading heading = new
                  HTMLHeading(1,"An HTML File List Example");
                heading.setAlign(HTMLConstants.CENTER);

                out.println(heading.getTag());
            }

            out.println("</body>\n");
            out.println("</html>\n");
            out.close();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }

    }
```

```
    /**
      *Process the POST request.
      *  @param req The request.
      *  @param res The response.
    **/
    public void doPost (HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
    {
        res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();

    }

    public void init(ServletConfig config)
    throws ServletException
    {
        super.init(config);

        // Create an AS400 object.
        sys_ = new AS400("mySystem", "myUID", "myPWD");
    }
}
```

範例：使用 HTMLTable 類別

下列範例將告訴您 HTMLTable 類別如何運作：

```
     // Create a default HTMLTable object.
      HTMLTable table = new HTMLTable();

     // Set the table attributes.
     table.setAlignment(HTMLTable.CENTER);
     table.setBorderWidth(1);

     // Create a default HTMLTableCaption object and set the caption text.
           HTMLTableCaption caption = new HTMLTableCaption();
     caption.setElement("Customer Account Balances - January 1, 2000");

     // Set the caption.
           table.setCaption(caption);

     // Create the table headers and add to the table.
     HTMLTableHeader account_header = new HTMLTableHeader(new HTMLText("ACCOUNT"));
     HTMLTableHeader name_header = new HTMLTableHeader(new HTMLText("NAME"));
     HTMLTableHeader balance_header = new HTMLTableHeader(new HTMLText("BALANCE"));

     table.addColumnHeader(account_header);
     table.addColumnHeader(name_header);
     table.addColumnHeader(balance_header);

     // Add rows to the table.  Each customer record represents a row in the table.
int numCols = 3;
     for (int rowIndex=0; rowIndex< numCustomers; rowIndex++)
     {
        HTMLTableRow row = new HTMLTableRow();
        row.setHorizontalAlignment(HTMLTableRow.CENTER);

           HTMLText account = new HTMLText(customers[rowIndex].getAccount());
           HTMLText name = new HTMLText(customers[rowIndex].getName());
           HTMLText balance = new HTMLText(customers[rowIndex].getBalance());

           row.addColumn(new HTMLTableCell(account));
           row.addColumn(new HTMLTableCell(name));
        row.addColumn(new HTMLTableCell(balance));
```

```
        // Add the row to the table.
      table.addRow(row);
}
System.out.println(table.getTag());
```

上述 Java 程式碼範例會產生下列 HTML 程式碼：

```
<table align="center" border="1">
<caption>Customer Account Balances - January 1, 2000</caption>
<tr>
<th>ACCOUNT</th>
<th>NAME</th>
<th>BALANCE</th>
</tr>
<tr align="center">
<td>0000001</td>
<td>Customer1</td>
<td>100.00</td>
</tr>
<tr align="center">
<td>0000002</td>
<td>Customer2</td>
<td>200.00</td>
</tr>
<tr align="center">
<td>0000003</td>
<td>Customer3</td>
<td>550.00</td>
</tr>
</table>
```

下表說明上面的 HTML 程式碼如何顯示在 Web 瀏覽器中：

表 2. Customer Account Balances - January 1, 2000

ACCOUNT	NAME	BALANCE
0000001	Customer1	100.00
0000002	Customer2	200.00
0000003	Customer3	550.00

範例：程式呼叫標記語言 (PCML)

下列範例使用「程式呼叫標記語言」來呼叫 i5/OS API，每一個範例都鏈結到一個顯示 Java 程式所遵循之 PCML 來源的頁面。

- 擷取資料的簡單範例：顯示擷取伺服器中使用者設定檔之相關資訊所需的 PCML 來源及 Java 程式。將呼叫的 API 是 *Retrieve User Information* (**QSYRUSRI**) API。
- 擷取資訊清單：顯示擷取伺服器中授權使用者清單所需的 PCML 來源及 Java 程式。 將呼叫的 API 是 *Open List of Authorized Users* (**QGYOLAUS**) API。此範例說明如何存取由伺服器程式傳回的結構陣列。
- 擷取多維度資料：顯示擷取「網路檔案系統 (NFS)」從伺服器匯出之清單所需的 PCML 來源及 Java 程式。將呼叫的 API 是 *Retrieve NFS Exports* (**QZNFRTVE**) API。這個範例描述如何存取結構陣列內的結構陣列。

註: 每一範例的適當權限雖有所不同，但可能包括特定物件權限及特殊權限。爲了執行這些範例，您用來登入的使用者設定檔，必須具有可執行下列作業的權限：

- 呼叫範例中的 i5/OS API
- 存取將要求的資訊

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：擷取資料的簡單範例

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

本範例包含兩部分：

- 呼叫 QSYRUSRI 的 PCML 來源
- 呼叫 QSYRUSRI 的 Java 程式原始碼

呼叫 QSYRUSRI 的 PCML 來源

```
<pcml version="1.0">

<!-- PCML source for calling "Retrieve user Information" (QSYRUSRI) API -->

  <!-- Format USRI0150 - Other formats are available -->
  <struct name="usri0100">
    <data name="bytesReturned"              type="int"    length="4"  usage="output"/>
    <data name="bytesAvailable"             type="int"    length="4"  usage="output"/>
    <data name="userProfile"                type="char"   length="10" usage="output"/>
    <data name="previousSignonDate"         type="char"   length="7"  usage="output"/>
    <data name="previousSignonTime"         type="char"   length="6"  usage="output"/>
    <data                                   type="byte"   length="1"  usage="output"/>
    <data name="badSignonAttempts"          type="int"    length="4"  usage="output"/>
    <data name="status"                     type="char"   length="10" usage="output"/>
    <data name="passwordChangeDate"         type="byte"   length="8"  usage="output"/>
    <data name="noPassword"                 type="char"   length="1"  usage="output"/>
    <data                                   type="byte"   length="1"  usage="output"/>
    <data name="passwordExpirationInterval" type="int"    length="4"  usage="output"/>
    <data name="datePasswordExpires"        type="byte"   length="8"  usage="output"/>
    <data name="daysUntilPasswordExpires"   type="int"    length="4"  usage="output"/>
    <data name="setPasswordToExpire"        type="char"   length="1"  usage="output"/>
    <data name="displaySignonInfo"          type="char"   length="10" usage="output"/>
  </struct>

  <!-- Program QSYRUSRI and its parameter list for retrieving USRI0100 format -->
  <program name="qsyrusri" path="/QSYS.lib/QSYRUSRI.pgm">
    <data name="receiver"         type="struct"   struct="usri0100"   usage="output"/>
    <data name="receiverLength"   type="int"      length="4"          usage="input" />
    <data name="format"           type="char"     length="8"          usage="input"    init="USRI0100"/>
    <data name="profileName"      type="char"     length="10"         usage="input"    init="*CURRENT"/>
    <data name="errorCode"        type="int"      length="4"          usage="input"    init="0"/>
  </program>

</pcml>
```

呼叫 QSYRUSRI 的 Java 程式原始碼

```
import com.ibm.as400.data.ProgramCallDocument;
import com.ibm.as400.data.PcmlException;
import com.ibm.as400.access.AS400;
import com.ibm.as400.access.AS400Message;
```

```
// Example program to call "Retrieve User Information" (QSYRUSRI) API
public class qsyrusri {

    public qsyrusri() {
    }

    public static void main(String[] argv)
    {
        AS400 as400System; // com.ibm.as400.access.AS400
        ProgramCallDocument pcml;// com.ibm.as400.data.ProgramCallDocument
        boolean rc = false;// Return code from ProgramCallDocument.callProgram()
        String msgId, msgText; // Messages returned from the server
        Object value;// Return value from ProgramCallDocument.getValue()

        System.setErr(System.out);

        // Construct AS400 without parameters, user will be prompted
        as400System = new AS400();

         try
        {
            // Uncomment the following to get debugging information
            //com.ibm.as400.data.PcmlMessageLog.setTraceEnabled(true);

            System.out.println("Beginning PCML Example..");
            System.out.println("    Constructing ProgramCallDocument for QSYRUSRI API...");

            // Construct ProgramCallDocument
            // First parameter is system to connect to
            // Second parameter is pcml resource name. In this example,
// serialized PCML file "qsyrusri.pcml.ser" or
            // PCML source file "qsyrusri.pcml" must be found in the classpath.
            pcml = new ProgramCallDocument(as400System, "qsyrusri");

            // Set input parameters. Several parameters have default values
            // specified in the PCML source. Do not need to set them using Java code.
System.out.println("    Setting input parameters...");
            pcml.setValue("qsyrusri.receiverLength",
                          new Integer((pcml.getOutputsize("qsyrusri.receiver"))));

            // Request to call the API
            // User will be prompted to sign on to the system
            System.out.println("    Calling QSYRUSRI API requesting information for the sign-on user.");
            rc = pcml.callProgram("qsyrusri");

            // If return code is false, we received messages from the server
            if (rc == false)
            {
                // Retrieve list of server messages
                AS400Message[] msgs = pcml.getMessageList("qsyrusri");

                // Iterate through messages and write them to standard output
                for (int m = 0; m < msgs.length; m++)
                {
                    msgId = msgs[m].getID();
                    msgText = msgs[m].getText();
                    System.out.println("    " + msgId + " - " + msgText);
                }
                System.out.println("** Call to QSYRUSRI failed. See messages above **");
                System.exit(0);
            }
            // Return code was true, call to QSYRUSRI succeeded
            // Write some of the results to standard output
            else
            {
                value = pcml.getValue("qsyrusri.receiver.bytesReturned");
```

```
                System.out.println("        Bytes returned:      " + value);
                value = pcml.getValue("qsyrusri.receiver.bytesAvailable");
                System.out.println("        Bytes available:     " + value);
                value = pcml.getValue("qsyrusri.receiver.userProfile");
                System.out.println("        Profile name:        " + value);
                value = pcml.getValue("qsyrusri.receiver.previousSignonDate");
                System.out.println("        Previous signon date:" + value);
                value = pcml.getValue("qsyrusri.receiver.previousSignonTime");
                System.out.println("        Previous signon time:" + value);
            }
        }
        catch(PcmlException e)
        {
            System.out.println(e.getLocalizedMessage());
            e.printStackTrace();
            System.out.println("*** Call to QSYRUSRI failed. ***");
            System.exit(0);
        }

        System.exit(0);
    } // End main()

}
```

範例：擷取資訊清單

本範例有兩部分：

- 呼叫 QGYOLAUS 的 PCML 來源
- 呼叫 QGYOLAUS 的 Java 程式原始碼

呼叫 QGYOLAUS 的 PCML 來源

```
<pcml version="1.0">

<!-- PCML source for calling "Open List of Authorized Users" (QGYOLAUS) API -->

  <!-- Format AUTU0150 - Other formats are available -->
  <struct name="autu0150">
    <data name="name"         type="char" length="10" />
    <data name="userOrGroup"  type="char" length="1"  />
    <data name="groupMembers" type="char" length="1"  />
    <data name="description"  type="char" length="50" />
  </struct>

  <!-- List information structure (common for "Open List" type APIs) -->
  <struct name="listInfo">
    <data name="totalRcds"    type="int"  length="4" />
    <data name="rcdsReturned" type="int"  length="4" />
    <data name="rqsHandle"    type="byte" length="4" />
    <data name="rcdLength"    type="int"  length="4" />
    <data name="infoComplete" type="char" length="1" />
    <data name="dateCreated"  type="char" length="7" />
    <data name="timeCreated"  type="char" length="6" />
    <data name="listStatus"   type="char" length="1" />
    <data                     type="byte" length="1" />
    <data name="lengthOfInfo" type="int"  length="4" />
    <data name="firstRecord"  type="int"  length="4" />
    <data                     type="byte" length="40" />
  </struct>

  <!-- Program QGYOLAUS and its parameter list for retrieving AUTU0150 format -->
  <program name="qgyolaus" path="/QSYS.lib/QGY.lib/QGYOLAUS.pgm" parseorder="listInfo receiver">
    <data   name="receiver"       type="struct" struct="autu0150" usage="output"
              count="listInfo.rcdsReturned" outputsize="receiverLength" />
    <data   name="receiverLength" type="int"    length="4"        usage="input" init="16384" />
```

```
    <data   name="listInfo"       type="struct" struct="listInfo" usage="output" />
    <data   name="rcdsToReturn"   type="int"    length="4"        usage="input" init="264" />
    <data   name="format"         type="char"   length="10"       usage="input" init="AUTU0150" />
    <data   name="selection"      type="char"   length="10"       usage="input" init="*USER" />
    <data   name="member"         type="char"   length="10"       usage="input" init="*NONE" />
    <data   name="errorCode"      type="int"    length="4"        usage="input" init="0" />

  </program>

  <!-- Program QGYGTLE returned additional "records" from the list
       created by QGYOLAUS. -->
  <program name="qgygtle" path="/QSYS.lib/QGY.lib/QGYGTLE.pgm" parseorder="listInfo receiver">
    <data   name="receiver"       type="struct" struct="autu0150" usage="output"
            count="listInfo.rcdsReturned" outputsize="receiverLength" />
    <data   name="receiverLength" type="int"    length="4"        usage="input" init="16384" />
    <data   name="requestHandle"  type="byte"   length="4"        usage="input" />
    <data   name="listInfo"       type="struct" struct="listInfo" usage="output" />
    <data   name="rcdsToReturn"   type="int"    length="4"        usage="input" init="264" />
    <data   name="startingRcd"    type="int"    length="4"        usage="input" />
    <data   name="errorCode"      type="int"    length="4"        usage="input" init="0" />
  </program>

  <!-- Program QGYCLST closes the list, freeing resources on the server -->
  <program name="qgyclst" path="/QSYS.lib/QGY.lib/QGYCLST.pgm" >
    <data   name="requestHandle"  type="byte"   length="4"        usage="input" />
    <data   name="errorCode"      type="int"    length="4"        usage="input" init="0" />
  </program>
</pcml>
```

呼叫 QGYOLAUS 的 Java 程式原始碼

```
import com.ibm.as400.data.ProgramCallDocument;
import com.ibm.as400.data.PcmlException;
import com.ibm.as400.access.AS400;
import com.ibm.as400.access.AS400Message;

// Example program to call "Retrieve List of Authorized Users" (QGYOLAUS) API
public class qgyolaus
{

  public static void main(String[] argv)
  {
    AS400 as400System; // com.ibm.as400.access.AS400
    ProgramCallDocument pcml;// com.ibm.as400.data.ProgramCallDocument
    boolean rc = false;// Return code from ProgramCallDocument.callProgram()
    String msgId, msgText; // Messages returned from the server
    Object value;// Return value from ProgramCallDocument.getValue()

    int[] indices = new int[1]; // Indices for access array value
    int nbrRcds,// Number of records returned from QGYOLAUS and QGYGTLE
        nbrUsers; // Total number of users retrieved
    String listStatus;// Status of list on the server
    byte[] requestHandle = new byte[4];

    System.setErr(System.out);

    // Construct AS400 without parameters, user will be prompted
    as400System = new AS400();

    try
    {
      // Uncomment the following to get debugging information
      //com.ibm.as400.data.PcmlMessageLog.setTraceEnabled(true);

      System.out.println("Beginning PCML Example..");
      System.out.println("    Constructing ProgramCallDocument for QGYOLAUS API...");
```

```
      // Construct ProgramCallDocument
      // First parameter is system to connect to
      // Second parameter is pcml resource name. In this example,
// serialized PCML file "qgyolaus.pcml.ser" or
      // PCML source file "qgyolaus.pcml" must be found in the classpath.
      pcml = new ProgramCallDocument(as400System, "qgyolaus");

      // All input parameters have default values specified in the PCML source.
      // Do not need to set them using Java code.

      // Request to call the API
      // User will be prompted to sign on to the system
      System.out.println("    Calling QGYOLAUS API requesting information for the sign-on user.");
      rc = pcml.callProgram("qgyolaus");

      // If return code is false, we received messages from the server
      if (rc == false)
      {
        // Retrieve list of server messages
        AS400Message[] msgs = pcml.getMessageList("qgyolaus");

        // Iterate through messages and write them to standard output
        for (int m = 0; m < msgs.length; m++)
        {
            msgId = msgs[m].getID();
            msgText = msgs[m].getText();
            System.out.println("    " + msgId + " - " + msgText);
        }
        System.out.println("** Call to QGYOLAUS failed. See messages above **");
        System.exit(0);
      }
      // Return code was true, call to QGYOLAUS succeeded
      // Write some of the results to standard output
      else
      {
        boolean doneProcessingList = false;
        String programName = "qgyolaus";
        nbrUsers = 0;
        while (!doneProcessingList)
        {
          nbrRcds = pcml.getIntValue(programName + ".listInfo.rcdsReturned");
          requestHandle = (byte[]) pcml.getValue(programName + ".listInfo.rqsHandle");

          // Iterate through list of users
          for (indices[0] = 0; indices[0] < nbrRcds; indices[0]++)
          {
               value = pcml.getValue(programName + ".receiver.name", indices);
               System.out.println("User:  " + value);

               value = pcml.getValue(programName + ".receiver.description", indices);
               System.out.println("\t\t" + value);
          }

          nbrUsers += nbrRcds;

          // See if we retrieved all the users.
          // If not, subsequent calls to "Get List Entries" (QGYGTLE)
          // would need to be made to retrieve the remaining users in the list.
          listStatus = (String) pcml.getValue(programName + ".listInfo.listStatus");
          if ( listStatus.equals("2")   // List is marked as "Complete"
            || listStatus.equals("3") ) // Or list is marked "Error building"
          {
            doneProcessingList = true;
          }
          else
          {
            programName = "qgygtle";
```

```
            // Set input parameters for QGYGTLE
            pcml.setValue("qgygtle.requestHandle", requestHandle);
            pcml.setIntValue("qgygtle.startingRcd", nbrUsers + 1);

            // Call "Get List Entries" (QGYGTLE) to get more users from list
            rc = pcml.callProgram("qgygtle");

            // If return code is false, we received messages from the server
            if (rc == false)
            {
              // Retrieve list of server messages
              AS400Message[] msgs = pcml.getMessageList("qgygtle");

              // Iterate through messages and write them to standard output
              for (int m = 0; m < msgs.length; m++)
              {
                  msgId = msgs[m].getID();
                  msgText = msgs[m].getText();
                  System.out.println("    " + msgId + " - " + msgText);
              }
              System.out.println("** Call to QGYGTLE failed. See messages above **");
              System.exit(0);
            }
            // Return code was true, call to QGYGTLE succeeded

          }
        }
        System.out.println("Number of users returned:  " + nbrUsers);

        // Call the "Close List" (QGYCLST) API
        pcml.setValue("qgyclst.requestHandle", requestHandle);
        rc = pcml.callProgram("qgyclst");
      }
    }
    catch(PcmlException e)
    {
      System.out.println(e.getLocalizedMessage());
      e.printStackTrace();
      System.out.println("*** Call to QGYOLAUS failed. ***");
      System.exit(0);
    }

    System.exit(0);
  }
}
```

範例：擷取多維度資料

本範例有兩部分：

- 呼叫 QZNFRTVE 的 PCML 來源
- 呼叫 QZNFRTVE 的 Java 程式原始碼

呼叫 QZNFRTVE 的 PCML 來源

```
<pcml version="1.0">

  <struct name="receiver">
    <data name="lengthOfEntry"            type="int"  length="4" />
    <data name="dispToObjectPathName"     type="int"  length="4" />
    <data name="lengthOfObjectPathName"   type="int"  length="4" />
    <data name="ccsidOfObjectPathName"    type="int"  length="4" />
    <data name="readOnlyFlag"             type="int"  length="4" />
    <data name="nosuidFlag"               type="int"  length="4" />
    <data name="dispToReadWriteHostNames" type="int"  length="4" />
```

```
  <data name="nbrOfReadWriteHostNames"  type="int"  length="4" />
  <data name="dispToRootHostNames"      type="int"  length="4" />
  <data name="nbrOfRootHostNames"       type="int"  length="4" />
  <data name="dispToAccessHostNames"    type="int"  length="4" />
  <data name="nbrOfAccessHostNames"     type="int"  length="4" />
  <data name="dispToHostOptions"        type="int"  length="4" />
  <data name="nbrOfHostOptions"         type="int"  length="4" />
  <data name="anonUserID"               type="int"  length="4" />
  <data name="anonUsrPrf"               type="char" length="10" />
  <data name="pathName"                 type="char" length="lengthOfObjectPathName"
        offset="dispToObjectPathName" offsetfrom="receiver" />

  <struct name="rwAccessList" count="nbrOfReadWriteHostNames"
          offset="dispToReadWriteHostNames" offsetfrom="receiver">
    <data name="lengthOfEntry"          type="int"  length="4" />
    <data name="lengthOfHostName"       type="int"  length="4" />
    <data name="hostName"               type="char" length="lengthOfHostName" />
    <data                               type="byte" length="0"
          offset="lengthOfEntry" />
  </struct>

  <struct name="rootAccessList" count="nbrOfRootHostNames"
          offset="dispToRootHostNames" offsetfrom="receiver">
    <data name="lengthOfEntry"          type="int"  length="4" />
    <data name="lengthOfHostName"       type="int"  length="4" />
    <data name="hostName"               type="char" length="lengthOfHostName" />
    <data                               type="byte" length="0"
          offset="lengthOfEntry" />
  </struct>

  <struct name="accessHostNames" count="nbrOfAccessHostNames"
          offset="dispToAccessHostNames" offsetfrom="receiver" >
    <data name="lengthOfEntry"          type="int"  length="4" />
    <data name="lengthOfHostName"       type="int"  length="4" />
    <data name="hostName"               type="char" length="lengthOfHostName" />
    <data                               type="byte" length="0"
          offset="lengthOfEntry" />
  </struct>

  <struct name="hostOptions" offset="dispToHostOptions" offsetfrom="receiver" count="nbrOfHostOptions">
    <data name="lengthOfEntry"          type="int"  length="4" />
    <data name="dataFileCodepage"       type="int"  length="4" />
    <data name="pathNameCodepage"       type="int"  length="4" />
    <data name="writeModeFlag"          type="int"  length="4" />
    <data name="lengthOfHostName"       type="int"  length="4" />
    <data name="hostName"               type="char" length="lengthOfHostName" />
    <data                               type="byte" length="0" offset="lengthOfEntry" />
  </struct>

  <data type="byte" length="0" offset="lengthOfEntry" />
</struct>

<struct name="returnedRcdsFdbkInfo">
  <data name="bytesReturned"            type="int" length="4" />
  <data name="bytesAvailable"           type="int" length="4" />
  <data name="nbrOfNFSExportEntries"    type="int" length="4" />
  <data name="handle"                   type="int" length="4" />
</struct>

<program name="qznfrtve" path="/QSYS.lib/QZNFRTVE.pgm" parseorder="returnedRcdsFdbkInfo receiver" >
  <data name="receiver"             type="struct" struct="receiver" usage="output"
           count="returnedRcdsFdbkInfo.nbrOfNFSExportEntries" outputsize="receiverLength"/>
  <data name="receiverLength"       type="int"    length="4" usage="input" init="4096" />
  <data name="returnedRcdsFdbkInfo" type="struct" struct="returnedRcdsFdbkInfo" usage="output" />
  <data name="formatName"           type="char"   length="8" usage="input" init="EXPE0100" />
  <data name="objectPathName"   type="char"   length="lengthObjPathName" usage="input" init="*FIRST" />
  <data name="lengthObjPathName"    type="int"    length="4" usage="input" init="6" />
```

```
    <data name="ccsidObjectPathName"  type="int"    length="4" usage="input" init="0" />
    <data name="desiredCCSID"         type="int"    length="4" usage="input" init="0" />
    <data name="handle"               type="int"    length="4" usage="input" init="0" />
    <data name="errorCode"            type="int"    length="4" usage="input" init="0" />
  </program>

</pcml>
```

呼叫 QZNFRTVE 的 Java 程式原始碼

```
import com.ibm.as400.data.ProgramCallDocument;
import com.ibm.as400.data.PcmlException;
import com.ibm.as400.access.AS400;
import com.ibm.as400.access.AS400Message;

// Example program to call "Retrieve NFS Exports" (QZNFRTVE) API
public class qznfrtve
{
  public static void main(String[] argv)
  {
    AS400 as400System; // com.ibm.as400.access.AS400
    ProgramCallDocument pcml;// com.ibm.as400.data.ProgramCallDocument
    boolean rc = false;// Return code from ProgramCallDocument.callProgram()
    String msgId, msgText; // Messages returned from the server
    Object value;// Return value from ProgramCallDocument.getValue()

    System.setErr(System.out);

    // Construct AS400 without parameters, user will be prompted
    as400System = new AS400();

    int[] indices = new int[2]; // Indices for access array value
    int nbrExports; // Number of exports returned
    int nbrOfReadWriteHostNames, nbrOfRWHostNames, nbrOfRootHostNames,
        nbrOfAccessHostnames,    nbrOfHostOpts;

    try
    {
      // Uncomment the following to get debugging information
      // com.ibm.as400.data.PcmlMessageLog.setTraceEnabled(true);

      System.out.println("Beginning PCML Example..");
      System.out.println("    Constructing ProgramCallDocument for QZNFRTVE API...");

      // Construct ProgramCallDocument
      // First parameter is system to connect to
      // Second parameter is pcml resource name. In this example,
// serialized PCML file "qznfrtve.pcml.ser" or
      // PCML source file "qznfrtve.pcml" must be found in the classpath.
      pcml = new ProgramCallDocument(as400System, "qznfrtve");

                              // Set input parameters.
Several parameters have default values
      // specified in the PCML source. Do not need to set them using Java code.
System.out.println("    Setting input parameters...");
      pcml.setValue("qznfrtve.receiverLength", new Integer( ( pcml.getOutputsize("qznfrtve.receiver"))));

      // Request to call the API
      // User will be prompted to sign on to the system
      System.out.println("    Calling QZNFRTVE API requesting NFS exports.");
      rc = pcml.callProgram("qznfrtve");

      if (rc == false)
      {
        // Retrieve list of server messages
        AS400Message[] msgs = pcml.getMessageList("qznfrtve");
```

```
    // Iterate through messages and write them to standard output
    for (int m = 0; m < msgs.length; m++)
    {
        msgId = msgs[m].getID();
        msgText = msgs[m].getText();
        System.out.println("    " + msgId + " - " + msgText);
    }
    System.out.println("** Call to QZNFRTVE failed. See messages above **");
    System.exit(0);
  }
  // Return code was true, call to QZNFRTVE succeeded
  // Write some of the results to standard output
  else
  {
    nbrExports = pcml.getIntValue("qznfrtve.returnedRcdsFdbkInfo.nbrOfNFSExportEntries");
    // Iterate through list of exports
    for (indices[0] = 0; indices[0] < nbrExports; indices[0]++)
    {
      value = pcml.getValue("qznfrtve.receiver.pathName", indices);
      System.out.println("Path name = " + value);

      // Iterate and write out Read Write Host Names for this export
      nbrOfReadWriteHostNames = pcml.getIntValue("qznfrtve.receiver.nbrOfReadWriteHostNames",
                                                 indices);
      for(indices[1] = 0; indices[1] < nbrOfReadWriteHostNames; indices[1]++)
      {
        value = pcml.getValue("qznfrtve.receiver.rwAccessList.hostName", indices);
        System.out.println("    Read/write access host name = " + value);
      }

      // Iterate and write out Root Host Names for this export
      nbrOfRootHostNames = pcml.getIntValue("qznfrtve.receiver.nbrOfRootHostNames", indices);
      for(indices[1] = 0; indices[1] < nbrOfRootHostNames; indices[1]++)
      {
        value = pcml.getValue("qznfrtve.receiver.rootAccessList.hostName", indices);
        System.out.println("    Root access host name = " + value);
      }

      // Iterate and write out Access Host Names for this export
      nbrOfAccessHostnames = pcml.getIntValue("qznfrtve.receiver.nbrOfAccessHostNames",
                                              indices);
      for(indices[1] = 0; indices[1] < nbrOfAccessHostnames; indices[1]++)
      {
        value = pcml.getValue("qznfrtve.receiver.accessHostNames.hostName", indices);
        System.out.println("    Access host name = " + value);
      }

      // Iterate and write out Host Options for this export
      nbrOfHostOpts = pcml.getIntValue("qznfrtve.receiver.nbrOfHostOptions", indices);
      for(indices[1] = 0; indices[1] < nbrOfHostOpts; indices[1]++)
      {
        System.out.println("    Host options:");
        value = pcml.getValue("qznfrtve.receiver.hostOptions.dataFileCodepage", indices);
        System.out.println("        Data file code page = " + value);
        value = pcml.getValue("qznfrtve.receiver.hostOptions.pathNameCodepage", indices);
        System.out.println("        Path name code page = " + value);
        value = pcml.getValue("qznfrtve.receiver.hostOptions.writeModeFlag", indices);
        System.out.println("        Write mode flag = " + value);
        value = pcml.getValue("qznfrtve.receiver.hostOptions.hostName", indices);
        System.out.println("        Host name = " + value);
      }
    } // end for loop iterating list of exports
  } // end call to QZNFRTVE succeeded
}
catch(PcmlException e)
{
  System.out.println(e.getLocalizedMessage());
```

```
      e.printStackTrace();
      System.exit(-1);
    }

    System.exit(0);
  } // end main()
}
```

範例：ReportWriter 類別

本節將列出 IBM Toolbox for Java ReportWriter 類別之文件所提供的所有程式碼範例。

JSPReportProcessor 及 PDFContext

- 範例：使用 JSPReportProcessor 與 PDFContext
- 範例：JSPReportProcessor 範例 JSP 檔

XSLReportProcessor 及 PCLContext

- 範例：使用 XSLReportProcessor 含 PCLContext
- 範例：XSLReportProcessor 範例 XML 檔
- 範例：XSLReportProcessor 範例 XSL 檔

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：使用 JSPReportProcessor 和 PDFContext

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

若要檢視可以與 JSPRunReport 搭配使用的範例 JSP 來源檔有哪些內容，請參閱 JSPcust_table.jsp。您也可以下載含有 JSP 範例檔的 ZIP 檔案。此壓縮檔也包含可以與 XSLReportProcessor 範例 (PCLRunReport) 搭配使用的 XML 和 XSL 範例檔。

```
////////////////////////////////////////////////////////////////////////////
//
// The following example (JSPRunReport) uses the JSPReportProcessor and the
// PDFContext classes to obtain data from a specified URL and convert the data
// to the PDF format. The data is then streamed to a file as a PDF document.
//
// Command syntax:
//     java JSPRunReport <jsp_Url> <output_filename>
//
////////////////////////////////////////////////////////////////////////////

import java.lang.*;
import java.awt.*;
import java.io.*;
import java.net.*;
```

```
import java.awt.print.*;
import java.awt.event.*;
import java.util.LinkedList;
import java.util.ListIterator;
import java.util.HashMap;

import com.ibm.xsl.composer.flo.*;
import com.ibm.xsl.composer.areas.*;
import com.ibm.xsl.composer.framework.*;
import com.ibm.xsl.composer.java2d.*;
import com.ibm.xsl.composer.prim.*;
import com.ibm.xsl.composer.properties.*;
import com.ibm.as400.util.reportwriter.processor.*;
import com.ibm.as400.util.reportwriter.pdfwriter.*;
import java.io.IOException;
import java.io.Serializable;
import org.xml.sax.SAXException;


public class JSPRunReport

{

    public static void main(String args[])
    {
        FileOutputStream fileout = null;

        /** specify the URL that contains the data you want to use in your report **/
        String JSPurl = args[0];
        URL jspurl = null;
              try {
              jspurl = new URL(JSPurl);
        }
        catch (MalformedURLException e)
        {}

        /** get output PDF file name**/
        String filename = args[1];
        try {
        fileout = new FileOutputStream(filename);
        }
        catch (FileNotFoundException e)
        {}

        /** set up page format **/
        Paper paper = new Paper();
        paper.setSize(612,792);
        paper.setImageableArea(18, 18, 576, 756);
        PageFormat pf = new PageFormat();
        pf.setPaper(paper);

        /** create a PDFContext object and cast FileOutputStream as an OutputStream **/
        PDFContext pdfcontext = new PDFContext((OutputStream)fileout, pf);

        System.out.println( Ready to parse XSL document );

        /** create the JSPReportProcessor object and set the template to the specified JSP **/
        JSPReportProcessor jspprocessor = new JSPReportProcessor(pdfcontext);
              try {
        jspprocessor.setTemplate(jspurl);
        }

        catch (NullPointerException np){
            String mes = np.getMessage();
            System.out.println(mes);
            System.exit(0);
            }
```

```
        /** process the report **/
        try {
        jspprocessor.processReport();
        }
        catch (IOException e) {
            String mes = e.getMessage();
            System.out.println(mes);
            System.exit(0);
            }
        catch (SAXException se) {
            String mes = se.getMessage();
            System.out.println(mes);
            System.exit(0);
            }

        System.exit(0);
    }
}
```

範例：JSPReportProcessor 範例 JSP 檔案

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
<?xml version="1.0"?>

<!--
  Copyright (c) 1999 The Apache Software Foundation.  All rights reserved.
-->

<%@ page session="false"%>
<%@ page language="java" contentType="text/html" %>
<%@ page import="java.lang.*" %>
<%@ page import="java.util.*" %>


<%-- <jsp:useBean id='cust_table' scope='page' class='table.JSPcust_table'/> --%>

<%!
    String[][] cust_data = new String [4][5];

    public void jspInit()
   {
      //cust_record_field [][] cust_data;
      // cust_record holds customer name, customer address, customer city, customer state,
     // customer zip


     String [] cust_record_1 = {"IBM","3602 4th St","Rochester","Mn","55901"};
     String [] cust_record_2 = {"HP","400 2nd","Springfield","Mo","33559"};
     String [] cust_record_3 = {"Wolzack","34 Hwy 52N","Lansing","Or","67895"};
     String [] cust_record_4 = {"Siems","343 60th","Salem","Tx","12345"};

      cust_data[0] = cust_record_1;
      cust_data[1] = cust_record_2;
      cust_data[2] = cust_record_3;
      cust_data[3] = cust_record_4;
   }
%>

<!-- First test of parse and compose. -->
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="theMaster" >
    <fo:region-body region-name="theRegion" margin-left=".2in"/>
    </fo:simple-page-master>
    <fo:page-sequence-master master-name="theMaster">
```

```
  <fo:single-page-master-reference master-name="thePage"/>
  </fo:page-sequence-master>
</fo:layout-master-set>
<fo:page-sequence master-name="theMaster">
 <fo:flow flow-name="theRegion">
   <fo:block>
      <fo:block text-align="center"> NORCAP </fo:block>
      <fo:block space-before=".2in" text-align="center">PAN PACIFIC HOTEL IN SAN FRANCISCO </fo:block>
      <fo:block text-align="center"> FRIDAY, DECEMBER 8-9, 2000 </fo:block>
   </fo:block>
   <fo:block space-before=".5in" font-size="8pt">
   <fo:table table-layout="fixed">
     <fo:table-column column-width="3in"/>
     <fo:table-column column-width="3in"/>
     <fo:table-column column-width="3in"/>
     <fo:table-column column-width="3in"/>
     <fo:table-column column-width="3in"/>
     <fo:table-body>
       <fo:table-row>
          <fo:table-cell column-number="1">
             <fo:block border-bottom-style="solid">NAME</fo:block>
          </fo:table-cell>
          <fo:table-cell column-number="2">
             <fo:block border-bottom-style="solid">ADDRESS</fo:block>
          </fo:table-cell>
          <fo:table-cell column-number="3">
             <fo:block border-bottom-style="solid">CITY</fo:block>
          </fo:table-cell>
          <fo:table-cell column-number="4">
             <fo:block border-bottom-style="solid">STATE</fo:block>
          </fo:table-cell>
          <fo:table-cell column-number="5">
             <fo:block border-bottom-style="solid">ZIP CODE</fo:block>
          </fo:table-cell>
       </fo:table-row>

       <%
         // add row to table
         for(int i = 0; i <= 3; i++)
         {
           String[] _array = cust_data[i];
       %>

       <fo:table-row>
         <fo:table-cell column-number="1">
           <fo:block space-before=".1in">
           <% if(_array[0].equals("IBM")) { %>
              <fo:inline background-color="blue">
              <% out.print(_array[0]); %>
              </fo:inline>
           <% } else { %>
              <% out.print(_array[0]); %>
           <% } %>
           </fo:block>
         </fo:table-cell>
         <fo:table-cell column-number="2">
           <fo:block space-before=".1in">
               <% out.print(_array[1]); %>
           </fo:block>
         </fo:table-cell>
         <fo:table-cell column-number="3">
           <fo:block space-before=".1in">
               <% out.print(_array[2]); %>
           </fo:block>
         </fo:table-cell>
         <fo:table-cell column-number="4">
           <fo:block space-before=".1in">
```

```
                <% out.print(_array[3]); %>
              </fo:block>
            </fo:table-cell>
            <fo:table-cell column-number="5">
             <fo:block space-before=".1in">
                <% out.print(_array[4]); %>
             </fo:block>
           </fo:table-cell>
          </fo:table-row>

       <%
       } // end row while
       %>

       </fo:table-body>
     </fo:table>
     </fo:block>
   </fo:flow>
  </fo:page-sequence>
</fo:root>
```

範例：使用 XSLReportProcessor 和 PCLContext

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// The following example (PCLRunReport) uses the XSLPReportProcessor and the
// PCLContext classes to obtain XML data and convert the data to the PCL format.
// The data is then streamed to a printer OutputQueue.
//
// To view the contents of example XML and XSL source files that you can use
// with PCLRunReport, see realestate.xml and realestate.xsl. You can also
// download a zip file that contains the XML and XSL example files. The zip
// file also contains a JSP example file that you can use with the
// JSPReportProcessor example (JSPRunReport).
//
// Command syntax:
//     java PCLRunReport <xml_file> <xsl_file>
//
////////////////////////////////////////////////////////////////////////////

import java.lang.*;
import java.awt.*;
import java.io.*;
import java.awt.print.*;
import java.awt.event.*;
import java.util.LinkedList;
import java.util.ListIterator;
import java.util.HashMap;

import com.ibm.xsl.composer.flo.*;
import com.ibm.xsl.composer.areas.*;
import com.ibm.xsl.composer.framework.*;
import com.ibm.xsl.composer.java2d.*;
import com.ibm.xsl.composer.prim.*;
import com.ibm.xsl.composer.properties.*;
import com.ibm.as400.util.reportwriter.processor.*;
import com.ibm.as400.util.reportwriter.pclwriter.*;
import java.io.IOException;
import java.io.Serializable;
import org.xml.sax.SAXException;
import com.ibm.as400.access.*;

public class PCLRunReport

{
```

```
public static void main(String args[])
{
    SpooledFileOutputStream fileout = null;
    String xmldocumentName = args[0];
    String xsldocumentName = args[1];

    String sys = "<system>";      /* Insert ISeries server name       */
    String user = "<user>";       /* Insert ISeries user profile name */
    String pass = "<password>";   /* Insert ISeries password          */

    AS400 system = new AS400(sys, user, pass);

    /* Insert ISeries output queue */
    String outqname = "/QSYS.LIB/qusrsys.LIB/<outq>.OUTQ";
    OutputQueue outq = new OutputQueue(system, outqname);
    PrintParameterList parms = new PrintParameterList();
    parms.setParameter(PrintObject.ATTR_OUTPUT_QUEUE, outq.getPath());

    try{
      fileout = new SpooledFileOutputStream(system, parms, null, null);
    }
    catch (Exception e)
    {}

    /** set up page format **/
    Paper paper = new Paper();
    paper.setSize(612,792);
    paper.setImageableArea(18, 36, 576, 720);
    PageFormat pf = new PageFormat();
    pf.setPaper(paper);

    /** create a PCLContext object and case FileOutputStream
        as an OutputStream **/
    PCLContext pclcontext = new PCLContext((OutputStream)fileout, pf);

    System.out.println("Ready to parse XSL document");

    /** create the XSLReportProcessor object **/
    XSLReportProcessor xslprocessor = new XSLReportProcessor(pclcontext);
    try {
    xslprocessor.setXMLDataSource(xmldocumentName);
    }
    catch (SAXException se) {
        String mes = se.getMessage();
        System.out.println(mes);
        System.exit(0);
        }
    catch (IOException ioe) {
        String mes = ioe.getMessage();
        System.out.println(mes);
        System.exit(0);
        }
    catch (NullPointerException np){
        String mes = np.getMessage();
        System.out.println(mes);
        System.exit(0);
        }
    /** set the template to the specified XML data source **/
    try {
    xslprocessor.setTemplate(xsldocumentName);
    }
    catch (NullPointerException np){
        String mes = np.getMessage();
        System.out.println(mes);
        System.exit(0);
        }
```

```
        catch (IOException e) {
            String mes = e.getMessage();
            System.out.println(mes);
            System.exit(0);
            }
        catch (SAXException se) {
            String mes = se.getMessage();
            System.out.println(mes);
            System.exit(0);
            }

        /** process the report **/
        try {
        xslprocessor.processReport();
        }
        catch (IOException e) {
            String mes = e.getMessage();
            System.out.println(mes);
            System.exit(0);
            }
        catch (SAXException se) {
            String mes = se.getMessage();
            System.out.println(mes);
            System.exit(0);
            }

        System.exit(0);
    }

}
```

範例：XSLReportProcessor 範例 XML 檔案

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
<?xml version="1.0"?>

<RESIDENTIAL-LISTINGS VERSION="061698">

<RESIDENTIAL-LISTING ID="ID1287" VERSION="061698">
  <GENERAL>
 <TYPE>Apartment</TYPE>
 <PRICE>$110,000</PRICE>
 <STRUCTURE><NUM-BEDS>3</NUM-BEDS><NUM-BATHS>1</NUM-BATHS></STRUCTURE>
  <AGE UNITS="YEARS">15</AGE>
  <LOCATION COUNTRY="USA" STATE="MA" COUNTY="MIDDLESEX" SECURITY="Public">
   <ADDRESS>13 Some Avenue</ADDRESS>
    <CITY>Dorchester</CITY><ZIP>02121</ZIP>
 </LOCATION>
  <IMAGE FORMAT="JPG" WIDTH="300" HEIGHT="150" SRC="house1.jpg"/>
 <MLS>
    <MLS-CODE SECURITY="Restricted">
   30224877
  </MLS-CODE>
    <MLS-SOURCE SECURITY="Public">
   <NAME>Bob the Realtor</NAME>
   <PHONE>1-617-555-1212</PHONE>
   <FAX>1-617-555-1313</FAX>
   <WEB>
    <EMAIL>Bob@bigbucks.com</EMAIL>
    <SITE>www.bigbucks.com</SITE>
   </WEB>
  </MLS-SOURCE>
 </MLS>
 <DATES><LISTING-DATE>3/5/98</LISTING-DATE></DATES>
 <LAND-AREA UNITS="ACRES">0.01</LAND-AREA>
```

```
 </GENERAL>

 <FEATURES>
    <DISCLOSURES>
 In your dreams.
    </DISCLOSURES>
 <UTILITIES>
Yes
 </UTILITIES>
 <EXTRAS>
Pest control included.
 </EXTRAS>
 <CONSTRUCTION>
Wallboard and glue
 </CONSTRUCTION>
 <ACCESS>
Front door.
 </ACCESS>
 </FEATURES>

 <FINANCIAL>
 <ASSUMABLE>
I assume so.
 </ASSUMABLE>
 <OWNER-CARRY>
Too heavy.
 </OWNER-CARRY>
 <ASSESMENTS>
    $150,000
 </ASSESMENTS>
 <DUES>
    $100
 </DUES>
 <TAXES>
    $2,000
 </TAXES>
 <LENDER>
Fly by nite mortgage co.
 </LENDER>
 <EARNEST>
Burt
 </EARNEST>
 <DIRECTIONS>
North, south, east, west
 </DIRECTIONS>
 </FINANCIAL>

 <REMARKS>
 </REMARKS>

 <CONTACTS>
<COMPANY>
 <NAME>
  Noplace Realty
 </NAME>
 <ADDRESS>
  12 Main Street
 </ADDRESS>
 <CITY>
  Lowell, MA
 </CITY>
 <ZIP>
  34567
 </ZIP>
</COMPANY>
<AGENT>
 <NAME>
```

```
   Mary Jones
  </NAME>
  <ADDRESS>
  </ADDRESS>
  <CITY>
  </CITY>
  <ZIP>
  </ZIP>
 </AGENT>
 <OWNER>
  <NAME>
  </NAME>
  <ADDRESS>
  </ADDRESS>
  <CITY>
  </CITY>
  <ZIP>
  </ZIP>
 </OWNER>
    <TENANT>
  Yes.
 </TENANT>
    <COMMISION>
  15%
 </COMMISION>
  </CONTACTS>

</RESIDENTIAL-LISTING>

<RESIDENTIAL-LISTING VERSION="061698" ID="ID1289">
  <GENERAL>

    <IMAGE FORMAT="JPG" WIDTH="300" HEIGHT="150" SRC="house2.jpg">
 </IMAGE>

 <MLS>
               <MLS-CODE SECURITY="Restricted">
      30298877
  </MLS-CODE>
               <MLS-SOURCE SECURITY="Public">
   <NAME>
    Mary the Realtor
   </NAME>
   <PHONE>
    1-617-555-3333
   </PHONE>
   <FAX>
    1-617-555-4444
   </FAX>
   <WEB>
    <EMAIL>
    Mary@somebucks.com
    </EMAIL>
    <SITE>
    www.bigbucks.com
    </SITE>
   </WEB>
  </MLS-SOURCE>
 </MLS>

 <TYPE>
  Home
 </TYPE>

 <PRICE>
  $200,000
 </PRICE>
```

```
       <AGE UNITS="MONTHS">
 3
</AGE>

       <LOCATION COUNTRY="USA" STATE="CO" COUNTY="MIDDLESEX" SECURITY="Public">
 <ADDRESS>
  1 Main Street
 </ADDRESS>
 <CITY>
 Boulder
 </CITY>
 <ZIP>
  11111
 </ZIP>
</LOCATION>

<STRUCTURE>
  <NUM-BEDS>
2
         </NUM-BEDS>
  <NUM-BATHS>
2
         </NUM-BATHS>
</STRUCTURE>

<DATES>
 <LISTING-DATE>
 4/3/98
               </LISTING-DATE>
</DATES>

       <LAND-AREA UNITS="ACRES">
 0.01
</LAND-AREA>

 </GENERAL>

 <FEATURES>
    <DISCLOSURES>
 In your dreams.
    </DISCLOSURES>
 <UTILITIES>
Yes
 </UTILITIES>
 <EXTRAS>
Pest control included.
 </EXTRAS>
 <CONSTRUCTION>
Wallboard and glue
 </CONSTRUCTION>
 <ACCESS>
Front door.
 </ACCESS>
 </FEATURES>

 <FINANCIAL>
 <ASSUMABLE>
I assume so.
 </ASSUMABLE>
 <OWNER-CARRY>
Too heavy.
 </OWNER-CARRY>
 <ASSESMENTS>
    $150,000
 </ASSESMENTS>
 <DUES>
```

```
    $100
  </DUES>
  <TAXES>
    $2,000
  </TAXES>
  <LENDER>
 Fly by nite mortgage co.
  </LENDER>
  <EARNEST>
 Burt
  </EARNEST>
  <DIRECTIONS>
 North, south, east, west
  </DIRECTIONS>
  </FINANCIAL>

  <REMARKS>
  </REMARKS>

  <CONTACTS>
 <COMPANY>
  <NAME>
   Noplace Realty
  </NAME>
  <ADDRESS>
   12 Main Street
  </ADDRESS>
  <CITY>
   Lowell, MA
  </CITY>
  <ZIP>
   34567
  </ZIP>
 </COMPANY>
 <AGENT>
  <NAME>
   Mary Jones
  </NAME>
  <ADDRESS>
  </ADDRESS>
  <CITY>
  </CITY>
  <ZIP>
  </ZIP>
 </AGENT>
 <OWNER>
  <NAME>
  </NAME>
  <ADDRESS>
  </ADDRESS>
  <CITY>
  </CITY>
  <ZIP>
  </ZIP>
 </OWNER>
    <TENANT>
  Yes.
 </TENANT>
    <COMMISION>
  15%
 </COMMISION>
  </CONTACTS>

</RESIDENTIAL-LISTING>
<RESIDENTIAL-LISTING VERSION="061698" ID="ID1290">
  <GENERAL>
```

```
   <IMAGE FORMAT="JPG" WIDTH="300" HEIGHT="150" SRC="house3.jpg">
</IMAGE>

<MLS>
               <MLS-CODE SECURITY="Restricted">
  20079877
 </MLS-CODE>
               <MLS-SOURCE SECURITY="Public">
  <NAME>
   Bob the Realtor
  </NAME>
  <PHONE>
   1-617-555-1212
  </PHONE>
  <FAX>
   1-617-555-1313
  </FAX>
  <WEB>
   <EMAIL>
   Bob@bigbucks.com
   </EMAIL>
   <SITE>
   www.bigbucks.com
   </SITE>
  </WEB>
 </MLS-SOURCE>
</MLS>

<TYPE>
 Apartment
</TYPE>

<PRICE>
 $65,000
</PRICE>

       <AGE UNITS="YEARS">
 30
</AGE>

       <LOCATION COUNTRY="USA" STATE="MA" COUNTY="MIDDLESEX" SECURITY="Public">
 <ADDRESS>
  25 Which Ave.
 </ADDRESS>
 <CITY>
 Cambridge
 </CITY>
 <ZIP>
  02139
 </ZIP>
</LOCATION>

<STRUCTURE>
  <NUM-BEDS>
3
         </NUM-BEDS>
  <NUM-BATHS>
1
         </NUM-BATHS>
</STRUCTURE>

<DATES>
 <LISTING-DATE>
 3/5/97
               </LISTING-DATE>
</DATES>
```

```
       <LAND-AREA UNITS="ACRES">
 0.05
</LAND-AREA>

 </GENERAL>

 <FEATURES>
    <DISCLOSURES>
 In your dreams.
    </DISCLOSURES>
 <UTILITIES>
Yes
 </UTILITIES>
 <EXTRAS>
Pest control included.
 </EXTRAS>
 <CONSTRUCTION>
Wallboard and glue
 </CONSTRUCTION>
 <ACCESS>
Front door.
 </ACCESS>
 </FEATURES>

 <FINANCIAL>
 <ASSUMABLE>
I assume so.
 </ASSUMABLE>
 <OWNER-CARRY>
Too heavy.
 </OWNER-CARRY>
 <ASSESMENTS>
    $150,000
 </ASSESMENTS>
 <DUES>
    $100
 </DUES>
 <TAXES>
    $2,000
 </TAXES>
 <LENDER>
Fly by nite mortgage co.
 </LENDER>
 <EARNEST>
Burt
 </EARNEST>
 <DIRECTIONS>
North, south, east, west
 </DIRECTIONS>
 </FINANCIAL>

 <REMARKS>
 </REMARKS>

 <CONTACTS>
<COMPANY>
 <NAME>
  Noplace Realty
 </NAME>
 <ADDRESS>
  12 Main Street
 </ADDRESS>
 <CITY>
  Lowell, MA
 </CITY>
 <ZIP>
  34567
```

```
 </ZIP>
</COMPANY>
<AGENT>
 <NAME>
  Mary Jones
 </NAME>
 <ADDRESS>
 </ADDRESS>
 <CITY>
 </CITY>
 <ZIP>
 </ZIP>
</AGENT>
<OWNER>
 <NAME>
 </NAME>
 <ADDRESS>
 </ADDRESS>
 <CITY>
 </CITY>
 <ZIP>
 </ZIP>
</OWNER>
   <TENANT>
 Yes.
</TENANT>
   <COMMISION>
 15%
</COMMISION>
 </CONTACTS>

</RESIDENTIAL-LISTING>
<RESIDENTIAL-LISTING VERSION="061698"  ID="ID1291">
  <GENERAL>

    <IMAGE FORMAT="JPG" WIDTH="300" HEIGHT="150" SRC="house4.jpg">
</IMAGE>

<MLS>
               <MLS-CODE SECURITY="Restricted">
  29389877
 </MLS-CODE>
               <MLS-SOURCE SECURITY="Public">
  <NAME>
   Mary the Realtor
  </NAME>
  <PHONE>
   1-617-555-3333
  </PHONE>
  <FAX>
   1-617-555-4444
  </FAX>
  <WEB>
   <EMAIL>
   Mary@somebucks.com
   </EMAIL>
   <SITE>
   www.bigbucks.com
   </SITE>
  </WEB>
 </MLS-SOURCE>
</MLS>

<TYPE>
       Home
</TYPE>
```

```
<PRICE>
 $449,000
</PRICE>

       <AGE UNITS="YEARS">
 7
</AGE>

       <LOCATION COUNTRY="USA" STATE="MA" COUNTY="MIDDLESEX" SECURITY="Public">
 <ADDRESS>
  100 Any Road
 </ADDRESS>
 <CITY>
 Lexington
 </CITY>
 <ZIP>
  02421
 </ZIP>
</LOCATION>

<STRUCTURE>
  <NUM-BEDS>
7
         </NUM-BEDS>
  <NUM-BATHS>
3
         </NUM-BATHS>
</STRUCTURE>

<DATES>
 <LISTING-DATE>
 6/8/98
               </LISTING-DATE>
</DATES>

       <LAND-AREA UNITS="ACRES">
 2.0
</LAND-AREA>

 </GENERAL>

 <FEATURES>
    <DISCLOSURES>
 In your dreams.
    </DISCLOSURES>
 <UTILITIES>
Yes
 </UTILITIES>
 <EXTRAS>
Pest control included.
 </EXTRAS>
 <CONSTRUCTION>
Wallboard and glue
 </CONSTRUCTION>
 <ACCESS>
Front door.
 </ACCESS>
 </FEATURES>

 <FINANCIAL>
 <ASSUMABLE>
I assume so.
 </ASSUMABLE>
 <OWNER-CARRY>
Too heavy.
 </OWNER-CARRY>
 <ASSESMENTS>
```

```
    $300,000
 </ASSESMENTS>
 <DUES>
    $100
 </DUES>
 <TAXES>
    $2,000
 </TAXES>
 <LENDER>
Fly by nite mortgage co.
 </LENDER>
 <EARNEST>
Burt
 </EARNEST>
 <DIRECTIONS>
North, south, east, west
 </DIRECTIONS>
 </FINANCIAL>

 <REMARKS>
 </REMARKS>

 <CONTACTS>
<COMPANY>
 <NAME>
  Noplace Realty
 </NAME>
 <ADDRESS>
  12 Main Street
 </ADDRESS>
 <CITY>
  Lowell, MA
 </CITY>
 <ZIP>
  34567
 </ZIP>
</COMPANY>
<AGENT>
 <NAME>
  Mary Jones
 </NAME>
 <ADDRESS>
 </ADDRESS>
 <CITY>
 </CITY>
 <ZIP>
 </ZIP>
</AGENT>
<OWNER>
 <NAME>
 </NAME>
 <ADDRESS>
 </ADDRESS>
 <CITY>
 </CITY>
 <ZIP>
 </ZIP>
</OWNER>
   <TENANT>
 Yes.
</TENANT>
   <COMMISION>
 15%
</COMMISION>
 </CONTACTS>
```

```
</RESIDENTIAL-LISTING>

</RESIDENTIAL-LISTINGS>
```

範例：XSLReportProcessor 範例 XSL 檔案

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
<?xml version="1.0"?>

<!-- Sample of styling an imagined real estate document. -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:fo="http://www.w3.org/1999/XSL/Format" >

  <xsl:template match="RESIDENTIAL-LISTINGS">
    <fo:root>
    <fo:layout-master-set>
 <fo:simple-page-master master-name="theMaster">
        <fo:region-body region-name="theRegion"/>
 </fo:simple-page-master>
  <fo:page-sequence-master master-name="theMaster">
    <fo:single-page-master-reference master-name="thePage" />
      </fo:page-sequence-master>
 </fo:layout-master-set>
  <fo:page-sequence master-name="theMaster">
  <fo:flow flow-name="theRegion">
          <xsl:apply-templates/>
 </fo:flow>
 </fo:page-sequence>
    </fo:root>
  </xsl:template>
  <xsl:template match="RESIDENTIAL-LISTING">


      <fo:block font-family="Times New Roman" font-weight="normal" font-size="24pt"
       background-color="silver" padding-before="5px" padding-after="5px"
       padding-start="5px" padding-end="5px" border-before-style="solid"
       border-before-color="blue" border-after-style="solid" border-after-color="blue"
       border-start-style="solid" border-start-color="blue" border-end-style="solid"
       border-end-color="blue">

 <fo:character character="y" background-color="blue" border-before-style="solid"
       border-before-color="yellow" border-after-style="solid" border-after-color="yellow"
       border-start-style="solid" border-start-color="yellow" border-end-style="solid"
       border-end-color="yellow" />
 </fo:block>


  </xsl:template>
</xsl:stylesheet>
```

範例：資源類別

本節將列出 IBM Toolbox for Java Resource 類別之文件所提供的所有程式碼範例。

Resource 與 ChangeableResource

- 範例：從 RUser (Resource 的具體子類別) 擷取屬性
- 範例：設定 RJob (ChangeableResource 的具體子類別) 的屬性值
- 範例：使用同屬程式碼來存取資源

ResourceList

- 範例：取得及列印 ResourceList 的內容
- 範例：使用同屬程式碼來存取 ResourceList
- 範例：在 servlet 中顯示資源清單

簡報

- 範例：使用簡報

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：資源清單

以下範例說明使用資源清單的各種方式：

- 範例：取得和列印 ResourceList 的內容
- 範例：使用同屬程式碼來存取 ResourceList
- 範例：在 servlet 中呈現資源清單

範例：取得和列印 ResourceList 的內容

ResourceList 具體次類別的一個範例就是 com.ibm.as400.resource.RJobList，它代表 iSeries 工作的清單。RJobList 支援許多選項 ID 和排序 ID，每一個都可以用來將清單加以過濾或排序。本範例列印出 RJobList 的內容：

```
    // Create an RJobList object to represent a list of jobs.
  AS400 system = new AS400("MYSYSTEM", "MYUSERID", "MYPASSWORD");
    RJobList jobList = new RJobList(system);

    // Filter the list to include only interactive jobs.
    jobList.setSelectionValue(RJobList.JOB_TYPE, RJob.JOB_TYPE_INTERACTIVE);

    // Sort the list by user name, then job name.
    Object[] sortValue = new Object[] { RJob.USER_NAME, RJob.JOB_NAME };
    jobList.setSortValue(sortValue);

    // Open the list and wait for it to complete.
    jobList.open();
    jobList.waitForComplete();

    // Read and print the contents of the list.
    long length = jobList.getListLength();
    for(long i = 0; i < length; ++i)
{
     System.out.println(jobList.resourceAt(i));
}

// Close the list.
jobList.close();
```

程式碼範例免責聲明

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：在 servlet 中顯示資源清單

請使用 ResourceListRowData 類別搭配 HTMLFormConverter 或 HTMLTableConverter 類別，即可在 servlet 中顯示資源清單。

- HTMLFormConverter 會把資源清單的內容顯示成一系列的格式，其中每份格式各包含有資源清單中的資源屬性值。
- HTMLTableConverter 會把資源清單的內容顯示成表格，每一列各包含有資源清單中的資源的相關資訊。

ResourceListRowData 物件用的直欄指定爲直欄屬性 ID 的陣列，每一列各代表資源物件。

```
    // Create the resource list.
This example creates
      // a list of all messages in the current user's message
      // queue.
    AS400 system = new AS400("MYSYSTEM", "MYUSERID", "MYPASSWORD");
      RMessageQueue messageQueue = new RMessageQueue(system, RMessageQueue.CURRENT);

      // Create the ResourceListRowData object.  In this example,
   // there are four columns in the table.  The first column
   // contains the icons and names for each message in the
      // message queue.  The remaining columns contain the text,
   // severity, and type for each message.
      ResourceListRowData rowdata = new ResourceListRowData(messageQueue,
       new Object[] { null, RQueuedMessage.MESSAGE_TEXT, RQueuedMessage.MESSAGE_SEVERITY,
                      RQueuedMessage.MESSAGE_TYPE } );

   // Create HTMLTable and HTMLTableConverter objects to
   // use for generating and customizing the HTML tables.
   HTMLTable table = new HTMLTable();
   table.setCellSpacing(6);
   table.setBorderWidth(8);

   HTMLTableConverter converter = new HTMLTableConverter();
   converter.setTable(table);
   converter.setUseMetaData(true);

   // Generate the HTML table.
   String[] html = converter.convert(rowdata);
   System.out.println(html[0]);
```

範例：從 Resource 擷取屬性值

Resource 的一個具體次類別是 com.ibm.as400.resource.RUser，它代表 iSeries 使用者。RUser 支援許多屬性 ID，每個 ID 都可以用來取得屬性值。

以下是從 RUser 擷取屬性值的範例：

```
    // Create an RUser object to refer to a specific user.
  AS400 system = new AS400("MYSYSTEM", "MYUSERID", "MYPASSWORD");
    RUser user = new RUser(system, "AUSERID");

    // Get the text description attribute value.
    String textDescription = (String)user.getAttributeValue(RUser.TEXT_DESCRIPTION);
```

範例：為 ChangeableResource 設定屬性值

ChangeableResource 的一個具體次類別是 com.ibm.as400.resource.RJob，它代表一項 iSeries 工作。RJob 支援許多屬性 ID，每個 ID 都可以用來存取屬性值。以下範例爲 RJob 設定兩個屬性值：

```
    // Create an RJob object to refer to a specific job.
  AS400 system = new AS400("MYSYSTEM", "MYUSERID", "MYPASSWORD");
    RJob job = new RJob(system, "AJOBNAME", "AUSERID", "AJOBNUMBER");

    // Set the date format attribute value.
    job.setAttributeValue(RJob.DATE_FORMAT, RJob.DATE_FORMAT_JULIAN);

    // Set the country or region ID attribute value.
    job.setAttributeValue(RJob.COUNTRY_ID, RJob.USER_PROFILE);

    // Commit both attribute changes.
    job.commitAttributeChanges();
```

範例：使用同屬程式碼來存取資源

您可以撰寫同屬程式碼來使用任何 Resource、ResourceList 或 ChangeableResource 子類別。這類程式碼可增進重複使用和維護的能力，而且不需經過修改，即可使用於未來的 Resource、ResourceList 或 ChangeableResource 子類別。

每個屬性都有相關的屬性 meta 資料物件 (com.ibm.as400.resource.ResourceMetaData)，可說明屬性的各種內容。這些內容包括屬性是否爲唯讀，以及預設和可能的值有哪些。

範例：

範例：列印 ResourceList 的內容

以下的同屬程式碼範例可列印出 ResourceList 的部分內容：

```
    void printContents(ResourceList resourceList, long numberOfItems) throws ResourceException
{
    // Open the list and wait for the requested number of items
    // to become available.
    resourceList.open();
    resourceList.waitForResource(numberOfItems);

    for(long i = 0; i < numberOfItems; ++i)
    {
        System.out.println(resourceList.resourceAt(i));
    }
}
```

範例：使用 ResourceMetaData 來存取資源所支援的每一個屬性

以下同屬程式碼範例可列印出資源支援的每一屬性的值。

```
void printAllAttributeValues(Resource resource) throws ResourceException
{
    // Get the attribute meta data.
    ResourceMetaData[] attributeMetaData = resource.getAttributeMetaData();
```

```
    // Loop through all attributes and print the values.
    for(int i = 0; i < attributeMetaData.length; ++i)
    {
        Object attributeID = attributeMetaData[i].getID();
        Object value = resource.getAttributeValue(attributeID);
        System.out.println("Attribute " + attributeID + " = " + value);
    }
}
```

範例：使用 ResourceMetaData 來重設 ChangeableResource 的每一個屬性

以下同屬程式碼範例可將 ChangeableResource 的所有屬性重設成爲預設值：

```
void resetAttributeValues(ChangeableResource resource) throws ResourceException
{
    // Get the attribute meta data.
    ResourceMetaData[] attributeMetaData = resource.getAttributeMetaData();

    // Loop through all attributes.
    for(int i = 0; i < attributeMetaData.length; ++i)
    {
        // If the attribute is changeable (not read only), then
        // reset its value to the default.
        if (! attributeMetaData[i].isReadOnly())
{
            Object attributeID = attributeMetaData[i].getID();
            Object defaultValue = attributeMetaData[i].getDefaultValue();
            resource.setAttributeValue(attributeID, defaultValue);
        }
    }

    // Commit all of the attribute changes.
    resource.commitAttributeChanges();
}
```

範例：RFML

本節將列出 IBM Toolbox for Java RFML 元件之文件所提供的所有程式碼範例：

- 範例：使用 RFML 與使用 IBM Toolbox for Java Record 類別兩者的比較
- 範例：RFML 來源檔

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：RFML 來源檔

以下的 RFML 來源檔範例可定義出 RFML 範例使用 RFML 與使用 IBM Toolbox for Java Record 類別兩者的比較中所使用的客戶記錄格式。這個 RFML 來源檔可以是命名爲 qcustcdt.rfml 的文字檔。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE rfml SYSTEM "rfml.dtd">

<rfml version="4.0" ccsid="819">

  <recordformat name="cusrec">

    <data name="cusnum" type="zoned" length="6"  precision="0" init="0"/>
    <data name="lstnam" type="char"  length="8"  ccsid="37" init="A"/>
    <data name="init"   type="char"  length="3"  ccsid="37" init="B"/>
    <data name="street" type="char"  length="13" ccsid="37" init="C"/>
    <data name="city"   type="char"  length="6"  ccsid="37" init="D"/>
    <data name="state"  type="char"  length="2"  ccsid="37" init="E"/>
    <data name="zipcod" type="zoned" length="5" init="1"/>
    <data name="cdtlmt" type="zoned" length="4" init="2"/>
    <data name="chgcod" type="zoned" length="1" init="3"/>
    <data name="baldue" type="zoned" length="6" precision="2" init="4"/>
    <data name="cdtdue" type="zoned" length="6" precision="2" init="5"/>

  </recordformat>

  <recordformat name="cusrec1">

    <data name="cusnum" type="zoned" length="6"  precision="0" init="0"/>
    <data name="lstnam" type="char"  length="8"  ccsid="37" init="A"/>
    <data name="init"   type="char"  length="3"  ccsid="37" init="B"/>
    <data name="street" type="char"  length="13" ccsid="37" init="C"/>
    <data name="city"   type="char"  length="6"  ccsid="37" init="D"/>
    <data name="state"  type="char"  length="2"  ccsid="37" init="E"/>
    <data name="zipcod" type="zoned" length="5" init="1"/>
    <data name="cdtlmt" type="zoned" length="4" init="2"/>
    <data name="chgcod" type="zoned" length="1" init="3"/>
    <data name="baldue" type="struct" struct="balance"/>
    <data name="cdtdue" type="struct" struct="balance"/>

  </recordformat>

  <recordformat name="cusrecAscii">

    <data name="cusnum" type="zoned" length="6"  precision="0" init="0"/>
    <data name="lstnam" type="char"  length="8" init="A"/>
    <data name="init"   type="char"  length="3" init="B"/>
    <data name="street" type="char"  length="13" init="C"/>
    <data name="city"   type="char"  length="6" init="D"/>
    <data name="state"  type="char"  length="2" init="E"/>
    <data name="zipcod" type="zoned" length="5" init="1"/>
    <data name="cdtlmt" type="zoned" length="4" init="2"/>
    <data name="chgcod" type="zoned" length="1" init="3"/>
    <data name="baldue" type="zoned" length="6" precision="2" init="4"/>
    <data name="cdtdue" type="zoned" length="6" precision="2" init="5"/>

  </recordformat>

  <struct name="balance">
    <data name="amount" type="zoned" length="6" precision="2" init="7"/>
  </struct>

</rfml>
```

範例：使用設定檔記號認證來交換 i5/OS 緒身分

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

下列程式碼範例將說明如何使用設定檔記號認證來交換 i5/OS 緒身分，並以特定使用者身分執行工作：

```
// Prepare to work with the local AS/400 system.
AS400 system = new AS400("localhost", "*CURRENT", "*CURRENT");

// Create a single-use ProfileTokenCredential with a 60 second timeout.
// A valid user ID and password must be substituted.
ProfileTokenCredential pt = new ProfileTokenCredential();
pt.setSystem(system);
pt.setTimeoutInterval(60);
pt.setTokenType(ProfileTokenCredential.TYPE_SINGLE_USE);

pt.setTokenExtended("USERID", "PASSWORD");

// Swap the i5/OS thread identity, retrieving a credential to
// swap back to the original identity later.
AS400Credential cr = pt.swap(true);

// Perform work under the swapped identity at this point.

// Swap back to the original i5/OS thread identity.
cr.swap();

// Clean up the credentials.
cr.destroy();
pt.destroy();
```

Servlet 類別範例

下面範例會顯示使用 Servlet 類別的一些方式：

- 範例：使用 ListRowData 類別
- 範例：使用 RecordListRowData 類別
- 範例：使用 SQLResultSetRowData 類別
- 範例：使用 HTMLFormConverter 類別
- 範例：使用 ListMetaData 類別
- 範例：使用 SQLResultSetMetaData 類別
- 範例：在 servlet 中呈現資源清單

您也可以將 Servlet 和 HTML 類別一起使用，如同本範例所示。

程式碼範例免責聲明

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：使用 ListRowData

本範例包含三個部分：

- 第 593 頁的『說明 ListRowData 類別如何運作的 Java 原始程式』

- 『使用 HTMLTableConverter 從 Java 原始程式產生的 HTML 原始程式』
- 第 594 頁的『瀏覽器如何顯示產生的 HTML』

說明 ListRowData 類別如何運作的 Java 原始程式

```
          // Access an existing non-empty data queue
KeyedDataQueue dq = new KeyedDataQueue(systemObject_, "/QSYS.LIB/MYLIB.LIB/MYDQ.DTAQ");

          // Create a metadata object.
          ListMetaData metaData = new ListMetaData(2);

          // Set first column to be the customer ID.
metaData.setColumnName(0, "Customer ID");
metaData.setColumnLabel(0, "Customer ID");
          metaData.setColumnType(0, RowMetaDataType.STRING_DATA_TYPE);

          // Set second column to be the order to be processed.
metaData.setColumnName(1, "Order Number");
metaData.setColumnLabel(1, "Order Number");
          metaData.setColumnType(1, RowMetaDataType.STRING_DATA_TYPE);

          // Create a ListRowData object.
          ListRowData rowData = new ListRowData();
          rowData.setMetaData(metaData);

          // Get the entries off the data queue.
KeyedDataQueueEntry data = dq.read(key, 0, "EQ");
 while (data != null)
{
   // Add queue entry to row data object.
   Object[] row = new Object[2];
   row[0] = new String(key);
   row[1] = new String(data.getData());
   rowData.addRow(row);

   // Get another entry from the queue.
   data = dq.read(key, 0, "EQ");
}

// Create an HTML converter object and convert the rowData to HTML.
HTMLTableConverter conv = new HTMLTableConverter();
conv.setUseMetaData(true);
HTMLTable[] html = conv.convertToTables(rowData);

// Display the output from the converter.
System.out.println(html[0]);
```

使用 HTMLTableConverter 從 Java 原始程式產生的 HTML 原始程式

使用上述 Java 原始程式範例中的第 219 頁的『HTMLTableConverter 類別』，會產生下列 HTML 程式碼。

```
<table>
<tr>
<th>Customer ID</th>
<th>Order Number</th>
</tr>
<tr>
<td>777-53-4444</td>
<td>12345-XYZ</td>
</tr>
<tr>
<td>777-53-4444</td>
<td>56789-ABC</td>
</tr>
</table>
```

瀏覽器如何顯示產生的 HTML

下表說明 HTML 原始程式碼在瀏覽器中看起來的樣子。

Customer ID	Order Number
777-53-4444	12345-XYZ
777-53-4444	56789-ABC

範例：使用 RecordListRowData

本範例包含三個部分：

- Java 原始程式，說明 RecordListRowData 類別如何運作
- HTML 原始程式，藉由使用 HTMLTableConverter，從 Java 原始程式產生
- 瀏覽器如何顯示所產生的 HTML

Java 原始程式，說明 RecordListRowData 類別如何運作

```
 // Create a server object.
AS400 mySystem = new AS400 ("mySystem.myComp.com", "UserId", "Password");

         // Get the path name for the file.
QSYSObjectPathName file = new QSYSObjectPathName(myLibrary, myFile, "%first%", "mbr");
         String ifspath = file.getPath();

         // Create a file object that represents the file.

         SequentialFile sf = new SequentialFile(mySystem, ifspath);

         // Retrieve the record format from the file.
AS400FileRecordDescription recordDescription = new AS400FileRecordDescription(mySystem, ifspath);
         RecordFormat recordFormat = recordDescription.retrieveRecordFormat()[0];

         // Set the record format for the file.
         sf.setRecordFormat(recordFormat);

         // Get the records in the file.
         Record[] records = sf.readAll();

         // Create a RecordListRowData object and add the records.
         RecordListRowData rowData = new RecordListRowData(recordFormat);

         for (int i=0; i < records.length; i++)
{
   rowData.addRow(records[i]);
}

// Create an HTML converter object and convert the rowData to HTML.
HTMLTableConverter conv = new HTMLTableConverter();
conv.setMaximumTableSize(3);
HTMLTable[] html = conv.convertToTables(rowData);

// Display the first HTML table generated by the converter.
System.out.println(html[0]);
```

HTML 原始程式，藉由使用 HTMLTableConverter，從 Java 原始程式產生

使用上述 Java 原始程式範例中的 HTMLTableConverter 類別，會產生下列 HTML 程式碼。

```
<table>
<tr>
<th>CNUM</th>
```

```
<th>LNAM</th>
<th>INIT</th>
<th>STR</th>
<th>CTY</th>
<th>STATE</th>
<th>ZIP</th>
<th>CTLMT</th>
<th>CHGCOD</th>
<th>BDUE</th>
<th>CTDUE</th>
</tr>
<tr>
<td>938472</td>
<td>Henning </td>
<td>G K</td>
<td>4859 Elm Ave </td>
<td>Dallas</td>
<td>TX</td>
<td align="right">75217</td>
<td align="right">5000</td>
<td align="right">3</td>
<td align="right">37.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>839283</td>
<td>Jones   </td>
<td>B D</td>
<td>21B NW 135 St</td>
<td>Clay  </td>
<td>NY</td>
<td align="right">13041</td>
<td align="right">400</td>
<td align="right">1</td>
<td align="right">100.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>392859</td>
<td>Vine    </td>
<td>S S</td>
<td>PO Box 79    </td>
<td>Broton</td>
<td>VT</td>
<td align="right">5046</td>
<td align="right">700</td>
<td align="right">1</td>
<td align="right">439.00</td>
<td align="right">0.00</td>
</tr>
</table>
```

瀏覽器如何顯示所產生的 HTML

下表說明 HTML 原始程式碼在瀏覽器中看起來的樣子。

CNUM	LNAM	INIT	STR	CTY	STATE	ZIP	CTLMT	CHGCOD	BDUE	CTDUE
938472	Henning	G K	4859 Elm Ave	Dallas	TX	75217	5000	3	37.00	0.00
839283	Jones	B D	21B NW 135 St	Clay	NY	13041	400	1	100.00	0.00
392859	Vine	S S	PO Box 79	Broton	VT	5046	700	1	439.00	0.00

範例：使用 SQLResultSetRowData

本範例包含三個部分：

- Java 原始程式，顯示 SQLResultSetRowData 類別運作的方式
- HTML 原始程式，藉由使用 HTMLTableConverter 從 Java 原始程式產生
- 瀏覽器如何顯示所產生的 HTML

Java 原始程式，可顯示 SQLResultSetRowData 類別運作的方式

```
        // Create a server object.
        AS400 mySystem = new AS400 ("mySystem.myComp.com", "UserId", "Password");

        // Register and get a connection to the database.

DriverManager.registerDriver (new com.ibm.as400.access.AS400JDBCDriver
());
        Connection connection = DriverManager.getConnection("jdbc:as400://" + mySystem.getSystemName());

        // Execute an SQL statement and get the result set.
        Statement statement = connection.createStatement();
        statement.execute("select * from qiws.qcustcdt");
        ResultSet resultSet = statement.getResultSet();

        // Create the SQLResultSetRowData object and initialize to the result set.
        SQLResultSetRowData rowData = new SQLResultSetRowData(resultSet);

        // Create an HTML table object to be used by the converter.
        HTMLTable table = new HTMLTable();

        // Set descriptive column headers.
        String[] headers = {"Customer Number", "Last Name", "Initials",
                                           "Street Address", "City", "State", "Zip Code",
                                           "Credit Limit", "Charge Code", "Balance Due",
                                           "Credit Due"};
        table.setHeader(headers);

        // Set several formatting options within the table.
        table.setBorderWidth(2);
        table.setCellSpacing(1);
        table.setCellPadding(1);

        // Create an HTML converter object and convert the rowData to HTML.
        HTMLTableConverter conv = new HTMLTableConverter();
        conv.setTable(table);
        HTMLTable[] html = conv.convertToTables(rowData);

        // Display the HTML table generated by the converter.
        System.out.println(html[0]);
```

HTML 原始程式，藉由使用 HTMLTableConverter 從 Java 原始程式產生

使用上述 Java 原始程式範例中的 HTMLTableConverter 類別，會產生下列 HTML 程式碼。

```
<table border="2" cellpadding="1" cellspacing="1">
<tr>
<th>Customer Number</th>
<th>Last Name</th>
<th>Initials</th>
<th>Street Address</th>
<th>City</th>
<th>State</th>
<th>Zip Code</th>
<th>Credit Limit</th>
<th>Charge Code</th>
```

```
<th>Balance Due</th>
<th>Credit Due</th>
</tr>
<tr>
<td>938472</td>
<td>Henning </td>
<td>G K</td>
<td>4859 Elm Ave </td>
<td>Dallas</td>
<td>TX</td>
<td align="right">75217</td>
<td align="right">5000</td>
<td align="right">3</td>
<td align="right">37.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>839283</td>
<td>Jones   </td>
<td
>B D</td>
<td>21B NW 135 St</td>
<td>Clay  </td>
<td>NY</td>
<td align="right">13041</td>
<td align="right">400</td>
<td align="right">1</td>
<td align="right">100.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>392859</td>
<td>Vine    </td>
<td>S S</td>
<td>PO Box 79    </td>
<td>Broton</td>
<td>VT</td>
<td align="right">5046</td>
<td align="right">700</td>
<td align="right">1</td>
<td align="right">439.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>938485</td>
<td>Johnson </td>
<td>J A</td>
<td>3 Alpine Way </td>
<td>Helen </td>
<td>GA</td>
<td align="right">30545</td>
<td align="right">9999</td>
<td align="right">2</td>
<td align="right">3987.50</td>
<td align="right">33.50</td>
</tr>
<tr>
<td>397267</td>
<td>Tyron   </td>
<td>W E</td>
<td>13 Myrtle Dr </td>
<td>Hector</td>
<td>NY</td>
<td align="right">14841</td>
<td align="right">1000</td>
<td align="right">1</td>
<td align="right">0.00</td>
```

```
<td align="right">0.00</td>
</tr>
<tr>
<td>389572</td>
<td>Stevens </td>
<td>K L</td>
<td>208 Snow Pass</td>
<td>Denver</td>
<td>CO</td>
<td align="right">80226</td>
<td align="right">400</td>
<td align="right">1</td>
<td align="right">58.75</td>
<td align="right">1.50</td>
</tr>
<tr>
<td>846283</td>
<td>Alison  </td>
<td>J S</td>
<td>787 Lake Dr  </td>
<td>Isle  </td>
<td>MN</td>
<td align="right">56342</td>
<td align="right">5000</td>
<td align="right">3</td>
<td align="right">10.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>475938</td>
<td>Doe     </td>
<td>J W</td>
<td>59 Archer Rd </td>
<td>Sutter</td>
<td>CA</td>
<td align="right">95685</td>
<td align="right">700</td>
<td align="right">2</td>
<td align="right">250.00</td>
<td align="right">100.00</td>
</tr>
<tr>
<td>693829</td>
<td>Thomas  </td>
<td>A N</td>
<td>3 Dove Circle</td>
<td>Casper</td>
<td>WY</td>
<td align="right">82609</td>
<td align="right">9999</td>
<td align="right">2</td>
<td align="right">0.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>593029</td>
<td>Williams</td>
<td>E D</td>
<td>485 SE 2 Ave </td>
<td>Dallas</td>
<td>TX</td>
<td align="right">75218</td>
<td align="right">200</td>
<td align="right">1</td>
<td align="right">25.00</td>
<td align="right">0.00</td>
</tr>
```

```
<tr>
<td>192837</td>
<td>Lee     </td>
<td>F L</td>
<td>5963 Oak St  </td>
<td>Hector</td>
<td>NY</td>
<td align="right">14841</td>
<td align="right">700</td>
<td align="right">2</td>
<td align="right">489.50</td>
<td align="right">0.50</td>
</tr>
<tr>
<td>583990</td>
<td>Abraham </td>
<td>M T</td>
<td>392 Mill St  </td>
<td>Isle  </td>
<td>MN</td>
<td align="right">56342</td>
<td align="right">9999</td>
<td align="right">3</td>
<td align="right">500.00</td>
<td align="right">0.00</td>
</tr>
</table>
```

瀏覽器如何顯示所產生的 HTML

下表說明 HTML 原始程式碼在瀏覽器中看起來的樣子。

客戶編號	姓氏	姓名縮寫	街道地址	城市	州	郵遞區號	信用額度	收費碼	待繳總額	待繳循環信用利息
938472	Henning	G K	4859 Elm Ave	Dallas	TX	75217	5000	3	37.00	0.00
839283	Jones	B D	21B NW 135 St	Clay	NY	13041	400	1	100.00	0.00
392859	Vine	S S	PO Box 79	Broton	VT	5046	700	1	439.00	0.00
938485	Johnson	J A	3 Alpine Way	Helen	GA	30545	9999	2	3987.50	33.50
397267	Tyron	W E	13 Myrtle Dr	Hector	NY	14841	1000	1	0.00	0.00
389572	Stevens	K L	208 Snow Pass	Denver	CO	80226	400	1	58.75	1.50
846283	Alison	J S	787 Lake Dr	Isle	MN	56342	5000	3	10.00	0.00
475938	Doe	J W	59 Archer Rd	Sutter	CA	95685	700	2	250.00	100.00
693829	Thomas	A N	3 Dove Circle	Casper	WY	82609	9999	2	0.00	0.00
593029	Williams	E D	485 SE 2 Ave	Dallas	TX	75218	200	1	25.00	0.00
192837	Lee	F L	5963 Oak St	Hector	NY	14841	700	2	489.50	0.50

客戶編號	姓氏	姓名縮寫	街道地址	城市	州	郵遞區號	信用額度	收費碼	待繳總額	待繳循環信用利息
583990	Abraham	M T	392 Mill St	Isle	MN	56342	9999	3	500.00	0.00

範例：使用 HTMLFormConverter

執行具有 servlet 支援的 web 伺服器時，請編譯及執行下列範例以瞭解 HTMLFormConverter 如何運作：

```
import java.awt.Color;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Enumeration;
import java.util.Hashtable;
import javax.servlet.*;
import javax.servlet.http.*;

import com.ibm.as400.util.html.GridLayoutFormPanel;
import com.ibm.as400.util.html.HTMLConstants;
import com.ibm.as400.util.html.HTMLForm;
import com.ibm.as400.util.html.HTMLTable;
import com.ibm.as400.util.html.HTMLTableCaption;
import com.ibm.as400.util.html.HTMLText;
import com.ibm.as400.util.html.LabelFormElement;
import com.ibm.as400.util.html.LineLayoutFormPanel;
import com.ibm.as400.util.html.SubmitFormInput;
import com.ibm.as400.util.html.TextFormInput;

import com.ibm.as400.util.servlet.HTMLFormConverter;
import com.ibm.as400.util.servlet.SQLResultSetRowData;

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.AS400JDBCDriver;

/**
  * 在 Servlet 中使用 HTMLFormConverter 類別的範例。
 */
public class HTMLFormConverterExample extends HttpServlet

{
   private String userId_   = "myUserId";
   private String password_ = "myPwd";
   private AS400  system_;

    private Connection databaseConnection_;



    // Perform cleanup before returning to the main HTML form.
    public void cleanup()
   {
       try
      {
          // Close the database connection.
         if (databaseConnection_ != null)
         {
            databaseConnection_.close();
            databaseConnection_ = null;
         }
      }
      catch (Exception e)
```

```
   {
      e.printStackTrace ();
   }
}


// Convert the row data to formatted HTML.
private HTMLTable[] convertRowData(SQLResultSetRowData rowData)
{
   try
   {
      // Create the converter, which will generate HTML from
      // the result set that comes back from the database query.
      HTMLFormConverter converter = new HTMLFormConverter();

      // Set the form attributes.
      converter.setBorderWidth(3);
      converter.setCellPadding(2);
      converter.setCellSpacing(4);

      // Convert the row data to HTML.
      HTMLTable[] htmlTable = converter.convertToForms(rowData);
      return htmlTable;
   }
   catch (Exception e)
   {
      e.printStackTrace ();
      return null;
   }

}


// Return the response to the client.
public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws IOException, ServletException
{
   response.setContentType("text/html");
   PrintWriter out = response.getWriter();

   out.println(showHtmlMain());
   out.close();
}


// Handle the data posted to the form.
public void doPost (HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException
{
   SQLResultSetRowData rowData = new SQLResultSetRowData();
   HTMLTable[] htmlTable = null;

   // Get the current session object or create one if needed.
   HttpSession session = request.getSession(true);

   ServletOutputStream out = response.getOutputStream();

   response.setContentType("text/html");

   Hashtable parameters = getRequestParameters (request);

   // Retrieve the row data and HTML table values for this session.
   rowData = (SQLResultSetRowData) session.getValue("sessionRowData");
   htmlTable = (HTMLTable[]) session.getValue("sessionHtmlTable");

   // if this is the first time through, show first record
   if (parameters.containsKey("getRecords"))
```

```
{
   rowData = getAllRecords(parameters, out);

   if (rowData != null)
   {
      // Set the row data value for this session.
      session.putValue("sessionRowData", rowData);

      // Position to the first record.
      rowData.first();

      // Convert the row data to formatted HTML.
      htmlTable = convertRowData(rowData);

      if (htmlTable != null)
      {
         rowData.first();
         session.putValue("sessionHtmlTable", htmlTable);
         out.println(showHtmlForRecord(htmlTable, 0));
      }
   }
}
// If the "Return To Main" button was pressed,
// go back to the main HTML form
else if (parameters.containsKey("returnToMain"))
{
   session.invalidate();
   cleanup();
   out.println(showHtmlMain());
}
// if the "First" button was pressed, show the first record
else if (parameters.containsKey("getFirstRecord"))
{
   rowData.first();
   out.println(showHtmlForRecord(htmlTable, 0));
}
// if the "Previous" button was pressed, show the previous record
else if (parameters.containsKey("getPreviousRecord"))
{
   if (!rowData.previous())
   {
      rowData.first();
   }
   out.println(showHtmlForRecord(htmlTable, rowData.getCurrentPosition()));
}
// if the "Next" button was pressed, show the next record
else if (parameters.containsKey("getNextRecord"))
{
   if (!rowData.next())
   {
      rowData.last();
   }
   out.println(showHtmlForRecord(htmlTable, rowData.getCurrentPosition()));
}
// if the "Last" button was pressed, show the last record
else if (parameters.containsKey("getLastRecord"))
{
   rowData.last();
   out.println(showHtmlForRecord(htmlTable, rowData.getCurrentPosition()));
}
// if none of the above, there must have been an error
else
{
   out.println(showHtmlForError("Internal error occurred. Unexpected parameters."));
}

// Save the row data value for this session so the current position
```

```
   // is updated in the object associated with this session.
   session.putValue("sessionRowData", rowData);

   // Close the output stream
   out.close();
}


// Get all the records from the file input by the user.
private SQLResultSetRowData getAllRecords(Hashtable parameters, ServletOutputStream out)
   throws IOException
{
   SQLResultSetRowData records = null;

   try
   {
      // Get the system, library and file name from the parameter list.
      String sys = ((String) parameters.get("System")).toUpperCase();
      String lib = ((String) parameters.get("Library")).toUpperCase();
      String file = ((String) parameters.get("File")).toUpperCase();
      if ((sys == null || sys.equals("")) ||
          (lib == null || lib.equals("")) ||
          (file == null || file.equals("")))
      {
         out.println(showHtmlForError("Invalid system, file or library name."));
      }
      else
      {
         // Get the connection to the server.
         getDatabaseConnection (sys, out);
         if (databaseConnection_ != null)
         {
            Statement sqlStatement = databaseConnection_.createStatement();

            // Query the database to get the result set.
            String query = "SELECT * FROM " + lib + "." + file;
            ResultSet rs = sqlStatement.executeQuery (query);

            boolean rsHasRows = rs.next();  // position cursor to first row

            // Show error message if the file contains no record;
            // otherwise, set row data to result set data.
            if (!rsHasRows)
            {
               out.println(showHtmlForError("No records in the file."));
            }
            else
            {
               records = new SQLResultSetRowData (rs);
            }

            // Don't close the Statement before we're done using
            // the ResultSet or bad things may happen.
            sqlStatement.close();
         }
      }
   }
   catch (Exception e)
   {
      e.printStackTrace ();
      out.println(showHtmlForError(e.toString()));
   }

   return records;
}
```

```
// Establish a database connection.
private void getDatabaseConnection (String sysName, ServletOutputStream out)
   throws IOException
{
   if (databaseConnection_ == null)
   {
      try
      {
         databaseConnection_ =
            DriverManager.getConnection("jdbc:as400://sysName,userId_,password_ );
      }
      catch (Exception e)
      {
         e.printStackTrace ();
         out.println(showHtmlForError(e.toString()));
      }
   }
}


// Gets the parameters from an HTTP servlet request.
private static Hashtable getRequestParameters (HttpServletRequest request)
{
   Hashtable parameters = new Hashtable ();
   Enumeration enum = request.getParameterNames();
   while (enum.hasMoreElements())
   {
      String key = (String) enum.nextElement();
      String value = request.getParameter (key);
      parameters.put (key, value);
   }
   return parameters;
}


// Get the servlet information.
public String getServletInfo()
{
   return "HTMLFormConverterExample";
}


// Perform initialization steps.
public void init(ServletConfig config)
{
   try
   {
      super.init(config);

      // Register the JDBC driver
      try
      {
         DriverManager.registerDriver(new AS400JDBCDriver());
      }
      catch (Exception e)
      {
         System.out.println("JDBC Driver not found");
      }
   }
   catch (Exception e)
   {
      e.printStackTrace();
   }
}


// Set the page header info.
```

```
private String showHeader(String title)
{
   StringBuffer page = new StringBuffer();
   page.append("<html><head><title>" + title + "</title>");
   page.append("</head><body bgcolor=\"blanchedalmond\">");
   return page.toString ();
}


// Show the HTML page with the appropriate error information.
private String showHtmlForError(String message)
{
   String title = "Error";
   StringBuffer page = new StringBuffer();
   page.append (showHeader (title));
   try
   {
      // Create the HTML Form object
      HTMLForm errorForm = new HTMLForm("HTMLFormConverterExample");

      // Set up so that doPost() gets called when the form is submitted.
      errorForm.setMethod(HTMLForm.METHOD_POST);

      // Create a single-column panel to which the
      // HTML elements will be added.
      GridLayoutFormPanel grid = new GridLayoutFormPanel();

      // Create the text element for the error and add it to the panel.
      HTMLText text = new HTMLText(message);
      text.setBold(true);
      text.setColor(Color.red);
      grid.addElement(text);

      // Create the button to return to main and add it to the panel.
      grid.addElement(new SubmitFormInput("returnToMain", "Return to Main"));

      // Add the panel to the HTML form.
      errorForm.addElement(grid);

      page.append(errorForm.toString());
   }
   catch (Exception e)
   {
      e.printStackTrace ();
   }
   page.append("</body></html>");
   return page.toString();
}


// Show the HTML form for an individual record.
private String showHtmlForRecord(HTMLTable[] htmlTable, int position)
{
   String title = "HTMLFormConverter Example";
   StringBuffer page = new StringBuffer();
   page.append (showHeader (title));

   page.append("<h1>" + title + "</h1>");

   try
   {
      // Create the HTML Form object
      HTMLForm recForm = new HTMLForm("HTMLFormConverterExample");

      // Set up so that doPost() gets called when the form is submitted.
      recForm.setMethod(HTMLForm.METHOD_POST);
```

```
      // Set up a single-column panel layout, within which to arrange
      // the generated HTML elements.
      GridLayoutFormPanel grid = new GridLayoutFormPanel();

      // Create and add a table caption that keeps track
      // of the current record.
      HTMLText recNumText = new HTMLText("Record number: " + (position + 1));
      recNumText.setBold(true);
      grid.addElement(recNumText);

      // Set up a two-column panel layout, within which to arrange
      // the table and text to comment on the converter output.
      GridLayoutFormPanel tableGrid = new GridLayoutFormPanel(2);
      tableGrid.addElement(htmlTable[position]);
      HTMLText comment = new HTMLText("   <---- Output from the HTMLFormConverter class");
      comment.setBold(true);
      comment.setColor(Color.blue);
      tableGrid.addElement(comment);

      // Add the table line to the panel.
      grid.addElement(tableGrid);

      // Set up a single-row panel layout, within which to arrange
      // the buttons for moving through the record list.
      LineLayoutFormPanel buttonLine = new LineLayoutFormPanel();
      buttonLine.addElement(new SubmitFormInput("getFirstRecord", "First"));
      buttonLine.addElement(new SubmitFormInput("getPreviousRecord", "Previous"));
      buttonLine.addElement(new SubmitFormInput("getNextRecord", "Next"));
      buttonLine.addElement(new SubmitFormInput("getLastRecord", "Last"));

      // Set up another single-row panel layout for the
      // Return To Main button.
      LineLayoutFormPanel returnToMainLine = new LineLayoutFormPanel();
      returnToMainLine.addElement(new SubmitFormInput("returnToMain", "Return to Main"));

      // Add the lines containing the buttons to the grid panel.
      grid.addElement(buttonLine);
      grid.addElement(returnToMainLine);

      // Add the panel to the form.
      recForm.addElement(grid);

      // Add the form to the HTML page.
      page.append(recForm.toString());
   }
   catch (Exception e)
   {
      e.printStackTrace ();
   }

   page.append("</body></html>");
   return page.toString();
}


// Show the main HTML form (request input for system, file,
// and library name).
private String showHtmlMain()
{
   String title = "HTMLFormConverter Example";
   StringBuffer page = new StringBuffer();
   page.append (showHeader (title));

   page.append("<h1>" + title + "</h1>");

   // Create the HTML Form object
   HTMLForm mainForm = new HTMLForm("HTMLFormConverterExample");
```

```
try
{
   // Set up so that doPost() gets called when the form is submitted.
   mainForm.setMethod(HTMLForm.METHOD_POST);

   // Add a brief description to the form.
   HTMLText desc =
      new HTMLText("<P>This example uses the HTMLFormConverter class " +
                   "to convert data retrieved from a server " +
                   "file.  The converter produces an array of HTML " +
                   "tables.  Each entry in the array is a record from " +
                   "the file.  " +
                   "Records are displayed one at a time, " +
                   "giving you buttons to move forward or backward " +
                   "through the list of records.</P>");
   mainForm.addElement(desc);

   // Add instructions to the form.
   HTMLText instr =
      new HTMLText("<P>Please input the name of the server, " +
                   "and the file and library name for the file you " +
                   "wish to access.  Then push the Show Records " +
                   "button to continue.</P>");
   mainForm.addElement(instr);


   // Create a grid layout panel and add the system, file
   // and library input fields.
   GridLayoutFormPanel panel = new GridLayoutFormPanel(2);

   LabelFormElement sysPrompt = new LabelFormElement("Server: ");
   TextFormInput system = new TextFormInput("System");
   system.setSize(10);

   LabelFormElement filePrompt = new LabelFormElement("File name: ");
   TextFormInput file = new TextFormInput("File");
   file.setSize(10);

   LabelFormElement libPrompt = new LabelFormElement("Library name: ");
   TextFormInput library = new TextFormInput("Library");
   library.setSize(10);

   panel.addElement(sysPrompt);
   panel.addElement(system);
   panel.addElement(filePrompt);
   panel.addElement(file);
   panel.addElement(libPrompt);
   panel.addElement(library);

   // Add the panel to the form.
   mainForm.addElement(panel);

   // Create the submit button and add it to the form.
   mainForm.addElement(new SubmitFormInput("getRecords", "Show Records"));
}
catch (Exception e)
{
   e.printStackTrace ();
}

page.append(mainForm.toString());
page.append("</body></html>");
```

```
      return page.toString();
   }

}
```

上述範例所產生的 HTML 看起來如下：

```
<table border="0">
<tr>
<td><b>Record number: 1</b></td>
</tr>
<tr>
<td><table border="0">
<tr>
<td><table border="3" cellpadding="2" cellspacing="4">
<tr>
<th>CUSNUM</th>
<td>839283</td>
</tr>
<tr>
<th>LSTNAM</th>
<td>Jones   </td>
</tr>
<tr>
<th>INIT</th>
<td>B D</td>
</tr>
<tr>
<th>STREET</th>
<td>21B NW 135 St</td>
</tr>
<tr>
<th>CITY</th>
<td>Clay  </td>
</tr>
<tr>
<th>STATE</th>
<td>NY</td>
</tr>
<tr>
<th>ZIPCOD</th>
<td>13041</td>
</tr>
<tr>
<th>CDTLMT</th>
<td>400</td>
</tr>
<tr>
<th>CHGCOD</th>
<td>1</td>
</tr>
<tr>
<th>BALDUE</th>
<td>100.00</td>
</tr>
<tr>
<th>CDTDUE</th>
<td>0.00</td>
</tr>
</table>
</td>
<td><font color="#0000ff"> <b><!-- Output from the HTMLFormConverter class-->
</b></font></td>
</tr>
</table>
</td>
</tr>
```

```
<tr>
<form>
<td><input type="submit" name="getFirstRecord" value="First" />
<input type="submit" name="getPreviousRecord" value="Previous" />
<input type="submit" name="getNextRecord" value="Next" />
<input type="submit" name="getLastRecord" value="Last" />
<br />
</td>
</tr>
<tr>
<td><input type="submit" name="returnToMain" value="Return to Main" />
<br />
</td>
</tr>
</table>
</form>
```

HTML 及 Servlet 類別的 Lights On 範例

此範例說明 HTML 與 servlet 類別如何運作。這是一般的概觀。如欲檢視此範例，請使用 Web 伺服器並執行瀏覽器以編譯與編譯本範例。

```
import java.io.IOException;
import java.io.CharArrayWriter;
import java.io.PrintWriter;
import java.sql.*;
import java.util.Enumeration;
import java.util.Hashtable;
import javax.servlet.*;
import javax.servlet.http.*;

import com.ibm.as400.util.html.*;
import com.ibm.as400.util.servlet.*;
import com.ibm.as400.access.*;

/*
在 Servlet 中使用 IBM Toolbox for Java 類別的範例。

伺服器中 SQL 資料庫綱目：

檔案 . . . . . .   LICENSES
檔案庫   . . .     LIGHTSON
欄位               類型             長度  空字元
 LICENSE           CHARACTER         10   NOT NULL
 USER_ID           CHARACTER         10   NOT NULL WITH DEFAULT
 E_MAIL            CHARACTER         20   NOT NULL
 WHEN_ADDED        DATE                   NOT NULL WITH DEFAULT
 TIME_STAMP        TIMESTAMP              NOT NULL WITH DEFAULT


檔案 . . . . . .   REPORTS
檔案庫   . . .     LIGHTSON
欄位               類型             長度  空字元
 LICENSE           CHARACTER         10   NOT NULL
 REPORTER          CHARACTER         10   NOT NULL WITH DEFAULT
 DATE_ADDED        DATE                   NOT NULL WITH DEFAULT
 TIME_ADDED        TIME                   NOT NULL WITH DEFAULT
 TIME_STAMP        TIMESTAMP              NOT NULL WITH DEFAULT
 LOCATION          CHARACTER         10   NOT NULL
 COLOR             CHARACTER         10   NOT NULL
 CATEGORY          CHARACTER         10   NOT NULL
*/

public class LightsOn extends javax.servlet.http.HttpServlet

{
```

```
private AS400  system_;
private String password_;  // password for the server and for the SQL database
private java.sql.Connection databaseConnection_;



public void destroy (ServletConfig config)
{
  try {
    if (databaseConnection_ != null) {
      databaseConnection_.close();
    }
  }
  catch (Exception e) { e.printStackTrace (); }
}

public void doGet (HttpServletRequest request,
                   HttpServletResponse response)
  throws IOException, ServletException
{
  HttpSession session = request.getSession();

  response.setContentType("text/html");
  PrintWriter out = response.getWriter();

  out.println(showHtmlMain());

  out.close();
}

public void doPost (HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
{
  HttpSession session = request.getSession(true);
  ServletOutputStream out = response.getOutputStream();
  response.setContentType("text/html");

  Hashtable parameters = getRequestParameters (request);

  if (parameters.containsKey("askingToReport"))
    out.println (showHtmlForReporting ());
  else if (parameters.containsKey("askingToRegister"))
    out.println (showHtmlForRegistering ());
  else if (parameters.containsKey("askingToUnregister"))
    out.println(showHtmlForUnregistering());
  else if (parameters.containsKey("askingToListRegistered"))
    out.println (showHtmlForListingAllRegistered ());
  else if (parameters.containsKey("askingToListReported"))
    out.println (showHtmlForListingAllReported ());
  else if (parameters.containsKey("returningToMain"))
    out.println (showHtmlMain ());


  else { // None of the above, so assume the user has filled out a form
         // and is submitting information.  Grab the incoming info
         // and do the requested action.

    if (parameters.containsKey("submittingReport")) {
      String acknowledgement = reportLightsOn (parameters, out);
      out.println (showAcknowledgement(acknowledgement));
    }

    else if (parameters.containsKey("submittingRegistration")) {
      String acknowledgement = registerLicense (parameters, out);
      out.println (showAcknowledgement(acknowledgement));
    }
```

```
    else if (parameters.containsKey("submittingUnregistration")) {
      String acknowledgement = unregisterLicense (parameters, out);
      out.println (showAcknowledgement(acknowledgement));
    }

    else {
      out.println (showAcknowledgement("Error (internal): " +
                          "Neither Report, Register, " +
                          "Unregister, ListRegistered, or ListReported."));
    }
  }

  out.close(); // Close the output stream.

}

// Gets the parameters from an HTTP servlet request, and packages them
// into a hashtable for convenience.
private static Hashtable getRequestParameters (HttpServletRequest request)
{
  Hashtable parameters = new Hashtable ();
  Enumeration enum = request.getParameterNames();
  while (enum.hasMoreElements()) {
    String key = (String) enum.nextElement();
    String value = request.getParameter (key);
    parameters.put (key, value);
  }
  return parameters;
}


// Removes blanks and hyphens from a String, and sets it to all uppercase.
private static String normalize (String oldString)
{
  if (oldString == null || oldString.length() == 0) return null;
  StringBuffer newString = new StringBuffer ();
  for (int i=0; i<oldString.length(); i++) {
    if (oldString.charAt(i) != ' ' && oldString.charAt(i) != '-')
      newString.append (oldString.charAt(i));
  }
  return newString.toString().toUpperCase();
}

// Composes a list of single-quoted strings.
private static String quoteList (String[] inList)
{
  StringBuffer outList = new StringBuffer();
  for (int i=0; i<inList.length; i++)
  {
    outList.append ("'" + inList[i] + "'");
    if (i<inList.length-1)
      outList.append (",");
  }
  return outList.toString();
}


public String getServletInfo ()
{
  return "Lights-On Servlet";
}

private AS400 getSystem ()
{
  try
  {
    if (system_ == null)
```

```
    {
      system_ = new AS400();

      // Note: It would be better to get these values
      // from a properties file.
      String sysName  = "MYSYSTEM";    // TBD
      String userId   = "MYUSERID";    // TBD
      String password = "MYPASSWD";    // TBD

      system_.setSystemName(sysName);
      system_.setUserId(userId);
      system_.setPassword(password);
      password_ = password;

      system_.connectService(AS400.DATABASE);
      system_.connectService(AS400.FILE);
      system_.addPasswordCacheEntry(sysName, userId, password_);
    }
  }
  catch (Exception e) { e.printStackTrace (); system_ = null; }

  return system_;
}


public void init (ServletConfig config)
{
  boolean rc;

  try {
    super.init(config);

    // Register the JDBC driver.
    try {
      java.sql.DriverManager.registerDriver(new com.ibm.as400.access.AS400JDBCDriver());
    }
    catch (Exception e)
    {
      System.out.println("JDBC Driver not found");
    }


  }
  catch (Exception e) { e.printStackTrace(); }
}

private void getDatabaseConnection ()
{
  if (databaseConnection_ == null) {
    try {
      databaseConnection_ = java.sql.DriverManager.getConnection(
                "jdbc:as400://" + getSystem().getSystemName() + "/" +
                "LIGHTSON", getSystem().getUserId(), password_ );

    }
    catch (Exception e) { e.printStackTrace (); }
  }
}


private String registerLicense (Hashtable parameters, ServletOutputStream out)
{
  String licenseNum = normalize ((String)parameters.get("licenseNum"));
  String eMailAddress = (String)parameters.get("eMailAddress");
  StringBuffer acknowledgement = new StringBuffer();

  if (licenseNum == null || licenseNum.length() == 0)
```

```
    acknowledgement.append ("Error: License number not specified.\n");

  if (eMailAddress == null || eMailAddress.length() == 0)
    acknowledgement.append ("Error: Notification e-mail address not specified.\n");

  if (acknowledgement.length() == 0)
  {
    try
    {
      // Insert the new license number and e-mail address into the database.
      getDatabaseConnection ();
      Statement sqlStatement = databaseConnection_.createStatement();

      // Issue the request.
      String cmd = "INSERT INTO LICENSES (LICENSE, E_MAIL) VALUES (" +
        quoteList(new String[] {licenseNum, eMailAddress}) + ")";
      sqlStatement.executeUpdate(cmd);
      sqlStatement.close();

      // Acknowledge the request.
      acknowledgement.append ("License number " + licenseNum + " has been registered.");
      acknowledgement.append ("Notification e-mail address is: " + eMailAddress);
    }
    catch (Exception e) { e.printStackTrace (); }
  }
  return acknowledgement.toString();
}


private String unregisterLicense (Hashtable parameters, ServletOutputStream out)
{
  String licenseNum = normalize ((String)parameters.get("licenseNum"));
  StringBuffer acknowledgement = new StringBuffer();

  if (licenseNum == null || licenseNum.length() == 0)
    acknowledgement.append ("Error: License number not specified.\n");

  if (acknowledgement.length() == 0)
  {
    try
    {
      // Remove the specified license number and e-mail address from database.
      getDatabaseConnection ();
      Statement sqlStatement = databaseConnection_.createStatement();

      // Delete the row(s) from the LICENSES database.
      String cmd = "DELETE FROM LICENSES WHERE LICENSE = '" + licenseNum + "'";
      sqlStatement.executeUpdate(cmd);
      sqlStatement.close();

      // Acknowledge the request.
      acknowledgement.append ("License number " + licenseNum + " has been unregistered.");
    }
    catch (Exception e) { e.printStackTrace (); }
  }
  return acknowledgement.toString();
}


private String reportLightsOn (Hashtable parameters, ServletOutputStream out)
{
  String licenseNum = normalize ((String)parameters.get("licenseNum"));
  String location = (String)parameters.get("location");
  String color = (String)parameters.get("color");
  String category = (String)parameters.get("category");
  StringBuffer acknowledgement = new StringBuffer();
  if (licenseNum == null || licenseNum.length() == 0)
```

```
    acknowledgement.append ("Error: License number not specified.");

  if (acknowledgement.length() == 0)
  {
    try
    {
      // Report "lights on" for a specified vehicle.
      getDatabaseConnection ();
      Statement sqlStatement = databaseConnection_.createStatement();

      // Add an entry to the REPORTS database.
      String cmd = "INSERT INTO REPORTS (LICENSE, LOCATION, COLOR, CATEGORY) VALUES (" +
        quoteList(new String[] {licenseNum, location, color, category}) + ")";
      sqlStatement.executeUpdate(cmd);
      sqlStatement.close();

      // Acknowledge the request.
      acknowledgement.append ("License number " + licenseNum + " has been reported.  Thanks!");
    }
    catch (Exception e) { e.printStackTrace (); }
  }
  return acknowledgement.toString();
}


private String showHeader (String title)
{
  StringBuffer page = new StringBuffer();
  page.append("<html><head><title>" + title + "</title>");
  page.append("</head><body bgcolor=\"blanchedalmond\">");
  return page.toString ();
}


private String showAcknowledgement (String acknowledgement)
{
  String title = "Acknowledgement";
  StringBuffer page = new StringBuffer();
  page.append (showHeader (title));

  try {
    HTMLForm form = new HTMLForm("LightsOn");
    GridLayoutFormPanel grid = new GridLayoutFormPanel();
    HTMLText text = new HTMLText(acknowledgement);
    if (acknowledgement.startsWith("Error"))
      text.setBold(true);
    grid.addElement(text);
    grid.addElement(new SubmitFormInput("returningToMain", "Home"));
    form.addElement(grid);
    page.append(form.toString());
  }
  catch (Exception e) { e.printStackTrace (); }
  page.append("</body></html>");
  return page.toString();
}


private String showHtmlMain ()
{
  String title = "Lights-On tool";
  StringBuffer page = new StringBuffer();
  page.append (showHeader (title));

  page.append("<h1>" + title + "</h1>");


  // Create the HTML Form object.
```

```
  HTMLForm mainForm = new HTMLForm("LightsOn");
  GridLayoutFormPanel grid = new GridLayoutFormPanel();

  try {
    // Set up so that doPost() gets called when the form is submitted.
    mainForm.setMethod(HTMLForm.METHOD_POST);

    // Create some buttons.
    grid.addElement(new SubmitFormInput("askingToReport", "Report a vehicle with lights on"));
    grid.addElement(new SubmitFormInput("askingToRegister", "Register my license number"));
    grid.addElement(new SubmitFormInput("askingToUnregister", "Unregister my license number"));
    grid.addElement(new SubmitFormInput("askingToListRegistered", "List all registered licenses"));
    grid.addElement(new SubmitFormInput("askingToListReported", "List all vehicles with lights on"));

    mainForm.addElement(grid);
  }
  catch (Exception e) { e.printStackTrace (); }

  page.append(mainForm.toString());
  page.append("</body></html>");

  return page.toString();
}


private String showHtmlForReporting ()
{
  String title = "Report a vehicle with lights on";
  StringBuffer page = new StringBuffer();
  page.append (showHeader (title));

  page.append("<h1>" + title + "</h1>");

  // Create the HTML Form object.
  HTMLForm reportForm = new HTMLForm("LightsOn");
  GridLayoutFormPanel grid = new GridLayoutFormPanel(2);

  try {
    // Set up so that doPost() gets called when the form is submitted.
    reportForm.setMethod(HTMLForm.METHOD_POST);

    TextFormInput licenseNum = new TextFormInput("licenseNum");
    licenseNum.setSize(10);
    licenseNum.setMaxLength(10);

    // Add elements to the line form
    grid.addElement(new LabelFormElement("Vehicle license number:"));
    grid.addElement(licenseNum);

    // Create a radio button group and add the radio buttons.
    RadioFormInputGroup colorGroup = new RadioFormInputGroup("color");

    colorGroup.add("color", "white", "white", true);
    colorGroup.add("color", "black", "black", false);
    colorGroup.add("color", "gray", "gray", false);
    colorGroup.add("color", "red", "red", false);
    colorGroup.add("color", "yellow", "yellow", false);
    colorGroup.add("color", "green", "green", false);
    colorGroup.add("color", "blue", "blue", false);
    colorGroup.add("color", "brown", "brown", false);

    // Create a selection list for category of vehicle.
    SelectFormElement category = new SelectFormElement("category");
    category.addOption("sedan", "sedan", true);
    category.addOption("convertible", "convertibl"); // 10-char field in DB
    category.addOption("truck", "truck");
    category.addOption("van", "van");
```

```
    category.addOption("SUV", "SUV");
    category.addOption("motorcycle", "motorcycle");
    category.addOption("other", "other");

    // Create a selection list for vehicle location (building number).
    SelectFormElement location = new SelectFormElement("location");
    location.addOption("001", "001", true);
    location.addOption("002", "002");
    location.addOption("003", "003");
    location.addOption("005", "005");
    location.addOption("006", "006");
    location.addOption("015", "015");

    grid.addElement(new LabelFormElement("Color:"));
    grid.addElement(colorGroup);

    grid.addElement(new LabelFormElement("Vehicle type:"));
    grid.addElement(category);

    grid.addElement(new LabelFormElement("Building:"));
    grid.addElement(location);

    grid.addElement(new SubmitFormInput("submittingReport", "Submit report"));
    grid.addElement(new SubmitFormInput("returningToMain", "Home"));

    reportForm.addElement(grid);
  }
  catch (Exception e) { e.printStackTrace (); }

  page.append(reportForm.toString());

  page.append("</body></html>");

  return page.toString();
}


private String showHtmlForRegistering ()
{
  String title = "Register my license number";
  StringBuffer page = new StringBuffer();
  page.append (showHeader (title));

  page.append("<h1>" + title + "</h1>");

  // Create the HTML Form object.
  HTMLForm registrationForm = new HTMLForm("LightsOn");

  // Set up a two-column panel layout, within which to arrange
  // the generated HTML elements.
  GridLayoutFormPanel grid = new GridLayoutFormPanel(2);

  try {
    // Set up so that doPost() gets called when the form is submitted.
    registrationForm.setMethod(HTMLForm.METHOD_POST);

    TextFormInput licenseNum = new TextFormInput("licenseNum");
    licenseNum.setSize(10);
    licenseNum.setMaxLength(10);

    TextFormInput eMailAddress = new TextFormInput("eMailAddress");
    eMailAddress.setMaxLength(20);

    grid.addElement(new LabelFormElement("License number:"));
    grid.addElement(licenseNum);

    grid.addElement(new LabelFormElement("E-mail notification address:"));
```

```
    grid.addElement(eMailAddress);

    grid.addElement(new SubmitFormInput("submittingRegistration", "Register"));
    grid.addElement(new SubmitFormInput("returningToMain", "Home"));

    registrationForm.addElement(grid);
  }
  catch (Exception e) { e.printStackTrace (); }

  page.append(registrationForm.toString());

  page.append("</body></html>");

  return page.toString();
}


private String showHtmlForUnregistering ()
{
  String title = "Unregister my license number";
  StringBuffer page = new StringBuffer();
  page.append (showHeader (title));

  page.append("<h1>" + title + "</h1>");

  // Create the HTML Form object.
  HTMLForm unregistrationForm = new HTMLForm("LightsOn");
  GridLayoutFormPanel grid = new GridLayoutFormPanel(2);

  try {
    // Set up so that doPost() gets called when the form is submitted.
    unregistrationForm.setMethod(HTMLForm.METHOD_POST);

    // Create the LineLayoutFormPanel object.
    TextFormInput licenseNum = new TextFormInput("licenseNum");
    licenseNum.setSize(10);
    licenseNum.setMaxLength(10);

    grid.addElement(new LabelFormElement("Vehicle license number:"));
    grid.addElement(licenseNum);

    grid.addElement(new SubmitFormInput("submittingUnregistration", "Unregister"));
    grid.addElement(new SubmitFormInput("returningToMain", "Home"));

    unregistrationForm.addElement(grid);
  }
  catch (Exception e) {
    e.printStackTrace ();
    CharArrayWriter cWriter = new CharArrayWriter();
    PrintWriter pWriter = new PrintWriter (cWriter, true);
    e.printStackTrace (pWriter);
    page.append (cWriter.toString());
  }

  page.append(unregistrationForm.toString());

  page.append("</body></html>");

  return page.toString();
}


private String showHtmlForListingAllRegistered ()
{
  String title = "All registered licenses";
  StringBuffer page = new StringBuffer();
  page.append (showHeader (title));
```

```
  try
  {
    // Create the HTML Form object.
    HTMLForm mainForm = new HTMLForm("LightsOn");

    // Set up a single-column panel layout, within which to arrange
    // the generated HTML elements.
    GridLayoutFormPanel grid = new GridLayoutFormPanel();

    // Specify the layout for the generated table.
    HTMLTable table = new HTMLTable();
    table.setAlignment(HTMLConstants.LEFT);
    table.setBorderWidth(3);

    // Create and add the table caption and header.
    HTMLTableCaption caption = new HTMLTableCaption();
    caption.setAlignment(HTMLConstants.TOP);
    caption.setElement(title);
    table.setCaption(caption);
    table.setHeader(new String[] { "License", "Date added" } );

    // Create the converter, which will generate table HTML from
    // the result set that comes back from the database query.
    HTMLTableConverter converter = new HTMLTableConverter();
    converter.setTable(table);

    getDatabaseConnection ();
    Statement sqlStatement = databaseConnection_.createStatement();

    // First pre-query the database to verify that it's not empty.
    String query = "SELECT COUNT(*) FROM LICENSES";
    ResultSet rs = sqlStatement.executeQuery (query);
    rs.next();  // position cursor to first row
    int rowCount = rs.getInt(1);

    if (rowCount == 0) {
      page.append ("<font size=4 color=red>No vehicles have been reported.</font>");
    }
    else {
      query = "SELECT LICENSE,WHEN_ADDED FROM LICENSES";
      rs = sqlStatement.executeQuery (query);
      SQLResultSetRowData rowData = new SQLResultSetRowData (rs);
      HTMLTable[] generatedHtml = converter.convertToTables(rowData);
      grid.addElement(generatedHtml[0]);
    }
    sqlStatement.close();
    // Note: Mustn't close statement before we're done using result set.

    grid.addElement(new SubmitFormInput("returningToMain", "Home"));

    mainForm.addElement(grid);
    page.append(mainForm.toString());
  }
  catch (Exception e) { e.printStackTrace (); }
  page.append("</body></html>");
  return page.toString();
}


private String showHtmlForListingAllReported ()
{
  String title = "All vehicles with lights on";
  StringBuffer page = new StringBuffer();
  page.append (showHeader (title));

  try
```

```
    {
      // Create the HTML Form object.
      HTMLForm form = new HTMLForm("LightsOn");

      // Set up a single-column panel layout, within which to arrange
      // the generated HTML elements.
      GridLayoutFormPanel grid = new GridLayoutFormPanel();

      // Specify the layout for the generated table.
      HTMLTable table = new HTMLTable();
      table.setAlignment(HTMLConstants.LEFT);
      table.setBorderWidth(3);

      // Create and add the table caption and header.
      HTMLTableCaption caption = new HTMLTableCaption();
      caption.setAlignment(HTMLConstants.TOP);
      caption.setElement(title);
      table.setCaption(caption);
      table.setHeader(new String[] { "License", "Color", "Category", "Date", "Time" } );

      // Create the converter, which will generate table HTML from
      // the result set that comes back from the database query.
      HTMLTableConverter converter = new HTMLTableConverter();
      converter.setTable(table);

      getDatabaseConnection ();
      Statement sqlStatement = databaseConnection_.createStatement();

      // First pre-query the database to verify that it's not empty.
      String query = "SELECT COUNT(*) FROM REPORTS";
      ResultSet rs = sqlStatement.executeQuery (query);
      rs.next();  // position cursor to first row
      int rowCount = rs.getInt(1);

      if (rowCount == 0) {
        page.append ("<font size=4 color=red>No vehicles have been reported.</font>");
      }
      else {
        query = "SELECT LICENSE,COLOR,CATEGORY,DATE_ADDED,TIME_ADDED FROM REPORTS";
        rs = sqlStatement.executeQuery (query);
        SQLResultSetRowData rowData = new SQLResultSetRowData (rs);
        HTMLTable[] generatedHtml = converter.convertToTables(rowData);
        grid.addElement(generatedHtml[0]);
      }
      sqlStatement.close();
      // Note: Mustn't close statement before we're done using result set.

      grid.addElement(new SubmitFormInput("returningToMain", "Home"));
      form.addElement(grid);
      page.append(form.toString());
    }
    catch (Exception e) { e.printStackTrace (); }
    page.append("</body></html>");
    return page.toString();
  }

}
```

簡式程式設計範例

這些範例說明您可以使用 IBM Toolbox for Java 類別，開始撰寫您自己的 Java 程式的一些方式。其用途在於供開始使用 IBM Toolbox for Java 類別的程式設計師參考，這些範例中包括程式碼中的關鍵行的詳細說明。

如果您需要入門協助，請參閱撰寫第一支 IBM Toolbox for Java 程式。

如需 IBM Toolbox for Java 資訊中提供之許多其他範例的鏈結，請參閱程式碼範例。

請使用下列清單來檢視簡式的程式設計範例：

- 呼叫指令
- 使用訊息佇列
- 使用記錄層次存取
- 使用 JDBC 類別建立及輸入表格資料
- 在 GUI 中顯示伺服器工作清單

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

撰寫第一支 IBM Toolbox for Java 程式

若要開始這個簡單的習題，您的工作站上必須已安裝 Java。您可以藉由複查執行 Java 應用程式的基本要求，來決定需要安裝哪一個版本。

將 Java 安裝到用戶端之後，請完成下列作業：

1. 將 jt400.jar 複製到工作站。
2. 將 JAR 檔的完整路徑附加到 CLASSPATH，以便將 jt400.jar 新增到 CLASSPATH。 例如，當 jt400.jar 檔位於您工作站 (執行 Windows) 上的 c:\lib 目錄時，請將下一行新增到 CLASSPATH 陳述式的尾端：

   ```
   ;c:\lib\jt400.jar
   ```
3. 開啓文字編輯器並鍵入 first simple programming example

   註: 請務必略過參照「附註」的文字 (例如，附註 1、附註 2 等)。將新文件儲存爲 CmdCall.java。
4. 在您的工作站上啓動一個指令階段作業，並使用下列指令編譯簡式程式設計範例：

   ```
   javac CmdCall.java
   ```
5. 在指令階段作業中，鍵入下列指令來執行簡式程式設計範例：

   ```
   java CmdCall
   ```

[簡式程式設計範例]

範例：使用 CommandCall

將下列內容當作程式的範例。

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
///////////////////////////////////////////////////////////////////////////
//
// Example using the IBM Toolbox for Java's access class, CommandCall.
```

```
//
// This source is an example of IBM Toolbox for Java "Job List".
//
//////////////////////////////////////////////////////////////////////////////
//
// The access classes of IBM Toolbox for Java are in the
// com.ibm.as400.access.package. Import this package to use the IBM Toolbox for
// Java classes.
//
//////////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;

public class CmdCall
{
    public static void main(String[] args)
   {
      // Like other Java classes, IBM Toolbox for Java classes
      // throw exceptions when something goes wrong.  These must
      // be caught by programs that use IBM Toolbox for Java.
      try 註 1
      {
         AS400 system = new AS400();

         CommandCall cc = new CommandCall(system); 註 2

         cc.run("CRTLIB MYLIB"); 註 3

         AS400Message[] ml = cc.getMessageList(); 註 4

          for (int i=0; i<ml.length; i++)
         {
            System.out.println(ml[i].getText()); 註 5
         }
      }
      catch (Exception e)
      {
         e.printStackTrace();
      }

           System.exit(0);
   }
}
```

1. IBM Toolbox for Java 使用 AS400 物件來識別目標伺服器。如果您建構 AS400 物件而未使用參數，則 IBM Toolbox for Java 會提示您輸入系統名稱、使用者 ID 及密碼。 AS400 類別也包括具有系統名稱、使用者 ID 與密碼的建構子。
2. 使用 IBM Toolbox for Java CommandCall 物件，將指令傳送至伺服器。在您建立 CommandCall 物件時，會同時將 AS400 傳入，因此 CommandCall 物件知道指令的目標伺服器爲何。
3. 使用 command call 物件的 run() 方法來執行指令。
4. 指令執行後會產生 i5/OS 訊息清單。IBM Toolbox for Java 會以 AS400Message 物件來呈現這些訊息。當指令完成時，您會看到 CommandCall 物件傳來的結果訊息。
5. 列印訊息文字。同時會傳回訊息 ID、訊息嚴重性以及其他資訊。此程式僅列印訊息文字。

範例：使用訊息佇列 (3 之 1)

[下一部分]

將下列內容當作程式的範例。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// Example using the Message Queue function of the IBM Toolbox for Java
//
// This source is an example of IBM Toolbox for Java "Message Queue".
//
////////////////////////////////////////////////////////////////////////////

package examples;  註 1


import java.io.*;
 import java.util.*;

import com.ibm.as400.access.*;  註 2

public class displayMessages extends Object
{

   public static void main(String[] parameters)  註 3
   {
      displayMessages me = new displayMessages();
      me.Main(parameters);  註 4

      System.exit(0);  註 5
   }


    void displayMessage()
   {
   }


    void Main(String[] parms)
   {
      try  註 6
      {

               // IBM Toolbox for Java code goes here
      }
      catch (Exception e)
      {
         e.printStackTrace();  註 7
      }
   }
}
```

1. 此類別位於 examples 套件中。Java 使用套件來避免 Java 類別檔案之間的名稱衝突。
2. 此指令行使得存取套件中的所有 IBM Toolbox for Java 類別都可以供本程式使用。存取套件中的類別都具有共同的字首 **com.ibm.as400**。藉由使用 import 陳述式，程式即可以它的名稱 (而非其完整的名稱) 來參照類別。 例如，您可以 AS400 來參照 AS400 類別，而不需要用 com.ibm.as400.AS400。
3. 此類別具有 **main** 方法；因此，它可以當作應用程式執行。若要呼叫程式，請執行 **java examples.displayMessages**。注意，執行程式時大小寫必須相符。由於使用了 IBM Toolbox for Java 類別，因此 jt400.zip 必須位於類別路徑環境變數中。
4. 附註 3 指出的 main 方法是靜態的。靜態方法的其中一項限制是它只能在其類別中呼叫其它靜態方法。爲避免這項限制，許多的 Java 程式會建立物件，然後在名爲 **Main** 的方法中進行起始設定處理程序。Main() 方法可以呼叫位於 displayMessages 物件中的其它方法。
5. IBM Toolbox for Java 可以代表應用程式建立一些緒，以執行 IBM Toolbox for Java 活動。如果程式沒有在結束時發出 **System.exit(0)**，就可能無法正常結束。例如，假設此程式是從 Windows 95 DOS 提示執行。若沒有此行，當程式完成時，就不會返回指令提示。使用者必須輸入 Ctrl-C 以取得指令提示。
6. IBM Toolbox for Java 程式碼會丟出程式必須捕捉的異常。

7. 此程式會在程式執行錯誤處理程序時，顯示異常文字。IBM Toolbox for Java 丟出的異常都已經過翻譯，所以異常文字會與工作站的語言相同。

[下一部分]

範例：使用訊息佇列 (3 之 2)

[上一部分 | 下一部分]

將下列內容當作程式的範例。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// Example using the Message Queue function of the IBM Toolbox for Java
//
// This source is an example of IBM Toolbox for Java "Message Queue".
//
////////////////////////////////////////////////////////////////////////////

package examples;


import java.io.*;
import java.util.*;
import com.ibm.as400.access.*;

public class displayMessages extends Object
{

   public static void main(String[] parameters)
   {
      displayMessages me = new displayMessages();

      me.Main(parameters);

      System.exit(0);
   }


   void displayMessage()
   {
   }


   void Main(String[] parms)
   {
      try
      {

          AS400 system = new AS400();  註 1


                     if (parms.length > 0)
                        system.setSystemName(parms[0]);  註 2

      }
      catch (Exception e)
      {
         e.printStackTrace();
      }
   }
}
```

1. 程式會使用 **AS400** 物件來指定要連接的伺服器。但是有一個例外，即需要伺服器資源的所有程式都必須擁有 AS/400 物件。此例外是 JDBC。 如果您的程式使用 JDBC，則 IBM Toolbox for Java JDBC 驅動程式會爲該程式建立 AS400 物件。
2. 此程式假設第一行指令行參數是伺服器的名稱。如果有傳遞參數給程式，則會使用 AS400 物件的 **setSystemName** 方法來設定系統名稱。AS400 物件也需要伺服器登入資訊：
   - 如果程式在工作站上執行，則 IBM Toolbox for Java 程式將提示使用者輸入使用者 ID 及密碼。**註：**如果未將系統名稱指定爲指令行參數，AS400 物件也會提示您提供系統名稱。
   - 如果程式在 iSeries JVM 上執行，則會使用執行 Java 程式之使用者的使用者 ID 及密碼。在此情況下，使用者不需指定系統名稱，但將系統名稱預設爲執行此程式所在的系統名稱。

[上一部分 | 下一部分]

範例：使用訊息佇列 (3 之 3)

[上個部分]

將下列內容當作程式的範例。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////////
//
// Example using the Message Queue function of the IBM Toolbox for Java
//
// This source is an example of IBM Toolbox for Java "Message Queue".
//
////////////////////////////////////////////////////////////////////////////////

package examples;


import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class displayMessages extends Object
{

   public static void main(String[] parameters)
   {
      displayMessages me = new displayMessages();

      me.Main(parameters);

      System.exit(0);
   }


   void displayMessage()
   {
   }


   void Main(String[] parms)
   {
      try
      {
         AS400 system = new AS400();

         if (parms.length > 0)
            system.setSystemName(parms[0]);
```

```
        MessageQueue queue = new MessageQueue(system, MessageQueue.CURRENT);  註 1


                Enumeration e = queue.getMessages();  註 2

                while (e.hasMoreElements())
                {

                    QueuedMessage message = (QueuedMessage) e.nextElement();  註 3
                    System.out.println(message.getText());  註 4
                 }
            }
            catch (Exception e)
            {
               e.printStackTrace();
            }
        }
}
```

1. 此程式的目的是將訊息顯示在伺服器訊息佇列中。這項作業會使用 IBM Toolbox for Java 的 **MessageQueue** 物件。建構了訊息佇列物件之後，參數即爲 AS400 物件和訊息佇登記稱。AS400 物件會指出哪一部伺服器有包含資源，而訊息佇登記稱會指定伺服器上的哪一個訊息佇列。在此狀況下將會使用常數，以便通知訊息佇列物件來存取登入使用者的佇列。
2. 訊息佇列物件將從伺服器中取得訊息的清單。此時會建立與伺服器的連線。
3. 從清單移除訊息。該訊息位於 IBM Toolbox for Java 程式的 QueuedMessage 物件中。
4. 列印訊息文字。

[上個部分]

範例：使用記錄層次存取 (2 之 1)

[下一部分]

將下列內容當作程式的範例。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////////
//
// Record level access example.
This program will prompt the user
// for the name of the server and the file to display.The file must exist
// and contain records.  Each record in the file will be displayed
// to System.out.
//
// Calling syntax: java RLSequentialAccessExample
//
// This source is an example of IBM Toolbox for Java "RecordLevelAccess"
//
///////////////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class RLSequentialAccessExample
{
   public static void main(String[] parameters)
   {
      BufferedReader inputStream = new BufferedReader(new InputStreamReader(System.in),1);
```

```
String systemName = "";
String library = "";
String file = "";
String member = "";

System.out.println();
 try
{
  System.out.print("System name: ");
  systemName = inputStream.readLine();

  System.out.print("Library in which the file exists: ");
  library = inputStream.readLine();

  System.out.print("File name: ");
  file = inputStream.readLine();

  System.out.print("Member name (press enter for first member): ");
  member = inputStream.readLine();
  if (member.equals(""))
  {
    member = "*FIRST";
  }

  System.out.println();
}
catch (Exception e)
{
  System.out.println("Error obtaining user input.");
  e.printStackTrace();
         System.exit(0);
}


AS400 system = new AS400(systemName);  註 1
 try
{
  system.connectService(AS400.RECORDACCESS);
}
catch (Exception e)
{
  System.out.println("Unable to connect for record level access.");
  System.out.println("Check the programmer's guide setup file for
    special instructions regarding record level access");
  e.printStackTrace();
  System.exit(0);
}


QSYSObjectPathName filePathName = new QSYSObjectPathName(library, file, member, "MBR");  註 2


SequentialFile theFile = new SequentialFile(system, filePathName.getPath());  註 3


AS400FileRecordDescription recordDescription =
     new AS400FileRecordDescription(system, filePathName.getPath());
try
{
  RecordFormat[] format = recordDescription.retrieveRecordFormat();  註 4


  theFile.setRecordFormat(format[0]);  註 5


  theFile.open(AS400File.READ_ONLY, 100, AS400File.COMMIT_LOCK_LEVEL_NONE);  註 6
```

```
        System.out.println("Displaying file " + library.toUpperCase() + "/" +
          file.toUpperCase() + "(" + theFile.getMemberName().trim() + "):");

        Record record = theFile.readNext();  註 7
        while (record != null)
        {
          System.out.println(record);
          record = theFile.readNext();
        }
        System.out.println();


        theFile.close();  註 8


        system.disconnectService(AS400.RECORDACCESS);  註 9
      }
      catch (Exception e)
      {
        System.out.println("Error occurred attempting to display the file.");
        e.printStackTrace();

         try
        {
          // Close the file
          theFile.close();
        }
        catch(Exception x)
        {
        }

        system.disconnectService(AS400.RECORDACCESS);
        System.exit(0);
      }

      // Make sure that the application ends; see readme for details
      System.exit(0);
    }
  }
```

1. 這一行程式碼會建立一個 AS400 物件，並連接到記錄層次存取服務。
2. 這個指令行會建立一個 QSYSObjectPathName 物件，用來取得要顯示的物件之整合檔案系統路徑名稱格式。
3. 此陳述式會建立一個物件，代表您所連接的伺服器上現有的循序檔。此循序檔即爲要顯示的檔案。
4. 這些指令行會擷取檔案的記錄格式。
5. 這一指令行會設定檔案的記錄格式。
6. 這一指令行會開啓所選取的檔案供讀取。可能的話，它一次會讀取 100 筆記錄。
7. 這行程式碼會循序讀取每一筆記錄。
8. 這一指令行會關閉檔案。
9. 這一指令行會切斷與記錄層次存取服務的連線。

[下一部分]

範例：使用記錄層次存取 (2 之 2)

[上個部分]

將下列內容當作程式的範例。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// Record level access example.
//
// Calling syntax: java RLACreateExample
//
////////////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class RLACreateExample
{
   public static void main(String[] args)
  {
    AS400 system = new AS400 (args[0]);

    String filePathName = "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/MBR1.MBR";  註 1

     try
    {
      SequentialFile theFile = new SequentialFile(system, filePathName);

      // Begin Note Two
      CharacterFieldDescription lastNameField =
         new CharacterFieldDescription(new AS400Text(20), "LNAME");
      CharacterFieldDescription firstNameField =
         new CharacterFieldDescription(new AS400Text(20), "FNAME");
      BinaryFieldDescription yearsOld =
         new BinaryFieldDescription(new AS400Bin4(), "AGE");

      RecordFormat fileFormat = new RecordFormat("RF");
      fileFormat.addFieldDescription(lastNameField);
      fileFormat.addFieldDescription(firstNameField);
      fileFormat.addFieldDescription(yearsOld);

      theFile.create(fileFormat, "A file of names and ages");  註 2
      // End Note Two


      theFile.open(AS400File.READ_WRITE, 1, AS400File.COMMIT_LOCK_LEVEL_NONE);

      // Begin Note Three
      Record newData = fileFormat.getNewRecord();
      newData.setField("LNAME", "Doe");
      newData.setField("FNAME", "John");
      newData.setField("AGE", new Integer(63));

      theFile.write(newData);  註 3
      // End Note Three

      theFile.close();
    }
    catch (Exception e)
    {
      System.out.println("An error has occurred: ");
      e.printStackTrace();
    }

    system.disconnectService(AS400.RECORDACCESS);

    System.exit(0);
  }
}
```

1. 上一行的 (args[0]) 和 MYFILE.FILE 是程式碼的片段，屬於其餘範例賴以執行的先決條件。該程式假設檔案庫 MYLIB 存在伺服器中，且使用者擁有它的存取權。
2. 在 Java 註解中，標示爲 Begin Note Two 與 End Note Two 之間的文字，可說明如何自行建立記錄格式，而非從現有的檔案取得記錄格式。此區塊的最後一行會在伺服器上建立檔案。
3. 在 Java 註解中，標示爲 Begin Note Three 與 End Note Three 之間的文字，可說明如何建立記錄，並予以寫入檔案中。

[上個部分]

範例：使用 JDBC 類別建立及輸入表格資料 (2 之 1)

[下個部分]

將下列內容當作程式的範例。

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
///////////////////////////////////////////////////////////////////////////////
//
// JDBCPopulate example.This program uses the IBM Toolbox for Java JDBC driver
// to create and populate a table.
//
// Command syntax:
//JDBCPopulate system collectionName tableName
//
// For example,
//JDBCPopulate MySystem MyLibrary MyTable
//
// This source is an example of IBM Toolbox for Java JDBC driver.
//
///////////////////////////////////////////////////////////////////////////////

import java.sql.*;

public class JDBCPopulate
{

    private static final String words[]
        = { "One",      "Two",      "Three",    "Four",     "Five",
            "Six",      "Seven",    "Eight",    "Nine",     "Ten",
            "Eleven",   "Twelve",   "Thirteen", "Fourteen", "Fifteen",
            "Sixteen",  "Seventeen","Eighteen", "Nineteen", "Twenty" };

    public static void main(String[] parameters)
    {

        if (parameters.length != 3) {
            System.out.println("");
            System.out.println("Usage:");
            System.out.println("");
            System.out.println("   JDBCPopulate system collectionName tableName");
            System.out.println("");
            System.out.println("");
            System.out.println("For example:");
            System.out.println("");
            System.out.println("");
            System.out.println("   JDBCPopulate MySystem MyLibrary MyTable");
            System.out.println("");
            return;
        }

        String system           = parameters[0];
        String collectionName   = parameters[1];
```

```
        String tableName        = parameters[2];

        Connection connection   = null;

        try {


            DriverManager.registerDriver(new com.ibm.as400.access.AS400JDBCDriver());  Note 1


            connection = DriverManager.getConnection ("jdbc:as400://"
                + system + "/" + collectionName);  Note 2

            try {
                Statement dropTable = connection.createStatement ();
                dropTable.executeUpdate ("DROP TABLE " + tableName);  附註 3
            }
            catch (SQLException e) {
            }


            Statement createTable = connection.createStatement ();
            createTable.executeUpdate ("CREATE TABLE " + tableName
                + " (I INTEGER, WORD VARCHAR(20), SQUARE INTEGER, "
                + " SQUAREROOT DOUBLE)");  附註 4


            PreparedStatement insert = connection.prepareStatement ("INSERT INTO "
                + tableName + " (I, WORD, SQUARE, SQUAREROOT) "
                + " VALUES (?, ?, ?, ?)");  Note 5


            for (int i = 1; i <= words.length; ++i) {
                insert.setInt (1, i);
                insert.setString (2, words[i-1]);
                insert.setInt (3, i*i);
                insert.setDouble (4, Math.sqrt(i));
                insert.executeUpdate ();  附註 6
            }

            System.out.println ("Table " + collectionName + "." + tableName
                + " has been populated.");
        }

        catch (Exception e) {
            System.out.println ();
            System.out.println ("ERROR: " + e.getMessage());
        }

        finally {

            try {
                if (connection != null)
                    connection.close ();  Note 7
            }
            catch (SQLException e) {
                // Ignore.
            }
        }

        System.exit (0);
    }

}
```

1. 此行會載入 IBM Toolbox for Java JDBC 驅動程式。JDBC 驅動程式是 JDBC 和您使用的資料庫之間的必要通訊管道。

2. 此陳述式會連接到資料庫。會出現提示請您提供使用者 ID 和密碼。系統會提供一個預設綱目，如此您就不需在 SQL 陳述式中定義表格名稱。
3. 這幾行會刪除表格 (如果存在的話)。
4. 這幾行會建立表格。
5. 這一行會準備將在表格中插入列的陳述式。由於您要執行這個陳述式數次，所以您應該使用 PreparedStatement 和參數標記。
6. 此程式碼區塊會輸入表格資料；每次執行迴圈時，它就會在表格中插入一列。
7. 此時表格已建立完成並填入資料，此陳述式會關閉與資料庫的連線。

[下個部分]

範例：使用 JDBC 類別建立及輸入表格資料 (2 之 2)

[上個部分]

將下列內容當作程式的範例。

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
///////////////////////////////////////////////////////////////////////////////
//
// JDBCQuery example.This program uses the IBM Toolbox for Java JDBC driver to
// query a table and output its contents.
//
// Command syntax:
//JDBCQuery system collectionName tableName
//
// For example,
//JDBCQuery MySystem qiws qcustcdt
//
// This source is an example of IBM Toolbox for Java JDBC driver.
//
///////////////////////////////////////////////////////////////////////////////

import java.sql.*;

public class JDBCQuery
{

    // Format a string so that it has the specified width.
    private static String format (String s, int width)
    {
        String formattedString;

        // The string is shorter than specified width,
        // so we need to pad with blanks.
        if (s.length() < width) {
            StringBuffer buffer = new StringBuffer (s);
            for (int i = s.length(); i < width; ++i)
                buffer.append (" ");
            formattedString = buffer.toString();
        }

        // Otherwise, we need to truncate the string.
        else
            formattedString = s.substring (0, width);

        return formattedString;
    }

    public static void main(String[] parameters)
```

```
    {
        // Check the input parameters.
        if (parameters.length != 3) {
            System.out.println("");
            System.out.println("Usage:");
            System.out.println("");
            System.out.println("   JDBCQuery system collectionName tableName");
            System.out.println("");
            System.out.println("");
            System.out.println("例如：");
            System.out.println("");
            System.out.println("");
            System.out.println("   JDBCQuery mySystem qiws qcustcdt");
            System.out.println("");
            return;
        }

        String system = parameters[0];
        String collectionName = parameters[1];
        String tableName= parameters[2];

        Connection connection = null;

        try {

            DriverManager.registerDriver(new com.ibm.as400.access.AS400JDBCDriver());  附註 1

            // Get a connection to the database.Since we do not
// provide a user id or password, a prompt will appear.
            connection = DriverManager.getConnection ("jdbc:as400://" + system);
            DatabaseMetaData dmd = connection.getMetaData ();  附註 2

            // Execute the query.
            Statement select = connection.createStatement ();
            ResultSet rs = select.executeQuery ("SELECT * FROM "
                + collectionName + dmd.getCatalogSeparator() + tableName);  附註 3

            // Get information about the result set.Set the column
// width to whichever is longer: the length of the label
            // or the length of the data.
            ResultSetMetaData rsmd = rs.getMetaData ();
            int columnCount = rsmd.getColumnCount ();  附註 4
            String[] columnLabels = new String[columnCount];
            int[] columnWidths = new int[columnCount];
            for (int i = 1; i <= columnCount; ++i) {
                columnLabels[i-1] = rsmd.getColumnLabel (i);
                columnWidths[i-1] = Math.max (columnLabels[i-1].length(),
                                              rsmd.getColumnDisplaySize (i));  附註 5
            }

            // Output the column headings.
            for (int i = 1; i <= columnCount; ++i) {
                System.out.print (format (rsmd.getColumnLabel(i), columnWidths[i-1]));
                System.out.print (" ");
            }
            System.out.println ();

            // Output a dashed line.
            StringBuffer dashedLine;
            for (int i = 1; i <= columnCount; ++i) {
                for (int j = 1; j <= columnWidths[i-1]; ++j)
                    System.out.print ("-");
                System.out.print (" ");
            }
            System.out.println ();

            // Iterate throught the rows in the result set and output
```

```
            // the columns for each row.
            while (rs.next ()) {
                for (int i = 1; i <= columnCount; ++i) {
                    String value = rs.getString (i);
                    if (rs.wasNull ())
                        value = "<null>";  附註 6
                    System.out.print (format (value, columnWidths[i-1]));
                    System.out.print (" ");
                }
                System.out.println ();
            }

        }

        catch (Exception e) {
            System.out.println ();
            System.out.println ("ERROR: " + e.getMessage());
        }

        finally {

            // Clean up.
            try {
                if (connection != null)
                    connection.close ();
            }
            catch (SQLException e) {
                // Ignore.
            }
        }

        System.exit (0);
    }

}
```

1. 此行會載入 IBM Toolbox for Java JDBC 驅動程式。JDBC 驅動程式扮演 JDBC 與您使用的資料庫之間的溝通管道。
2. 這一行會擷取連線的 meta 資料，該資料是說明資料庫諸多特性的物件。
3. 此陳述式會對指定的表格執行查詢。
4. 這幾行會擷取有關表格的資訊。
5. 這幾行會將直欄寬度設爲標籤長度或資料長度 (以較長者爲準)。
6. 此程式碼區塊會疊代表格中的所有列，並顯示每一列中每個直欄的內容。

[上個部分]

範例：以 GUI 形式顯示伺服器工作清單

將下列內容當作程式的範例。

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
///////////////////////////////////////////////////////////////////////////////
//
// Example using the IBM Toolbox for Java's vaccess
// class, VJobList.
//
// This source is an example of IBM Toolbox for Java "Job List".
//
///////////////////////////////////////////////////////////////////////////////

package examples;  註 1
```

```
import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;  註 2

import javax.swing.*;  註 3
import java.awt.*;
import java.awt.event.*;

public class GUIExample
{

  public static void main(String[] parameters)  註 4
  {
      GUIExample example = new GUIExample(parameters);

  }

  public GUIExample(String[] parameters)
  {
     try  註 5
     {
           // Create an AS400 object.
      //The system name was passed as the first command line argument.
          AS400 system = new AS400 (parameters[0]);  註 6

          VJobList jobList = new VJobList (system);  註 7

     // Create a frame.
     JFrame frame = new JFrame ("Job List Example");  註 8

     // Create an error dialog adapter.  This will display any errors to the user.
     ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (frame);  註 9

     // Create an explorer pane to present the job list.
     AS400ExplorerPane explorerPane = new AS400ExplorerPane (jobList);  註 10

     explorerPane.addErrorListener (errorHandler);  註 11

     // Use load to load the information from the system.

     explorerPane.load();  註 12

     // When the frame closes, exit the program.
     frame.addWindowListener (new WindowAdapter ()  註 13
     {
           public void windowClosing (WindowEvent event)
           {
              System.exit(0);
           }
     } );

     // Layout the frame with the explorer pane.
     frame.getContentPane().setLayout(new BorderLayout() );
     frame.getContentPane().add("Center", explorerPane);  註 14

     frame.pack();
     frame.show();  註 15
     }

     catch (Exception e)
     {
          e.printStackTrace();  註 16
     }
     System.exit(0);  註 17

     }
}
```

1. 此類別位於範例套件中。Java 使用套件來避免 Java 類別檔案之間的名稱衝突。
2. 此指令行使得 Vaccess 套件中的所有 IBM Toolbox for Java 類別都可供本程式使用。Vaccess 套件中的類別具有通用字首 com.ibm.as400.vaccess。藉由使用 import 陳述式，程式會呼叫名稱，而不是呼叫套件加上名稱。例如，您可使用 AS400ExplorerPane，而非 com.ibm.as400.AS400ExplorerPane 參照 AS400ExplorerPane 類別。
3. 此指令行使得 Swing 套件中的所有「Java 基礎類別 (JFC)」都可供本程式使用。若要使用 IBM Toolbox for Java Vaccess (GUI) 類別的 Java 程式，必須具備 Sun Microsystems 公司的 JDK 1.1.2 與 Java Swing 1.0.3。Swing 可從 Sun 的 JFC 1.1 中取得。
4. 此類別具有 main 方法，因此可當作應用程式執行。若要呼叫此程式，請執行 java examples.GUIExample serverName，其中 serverName 是您的伺服器名稱。jt400.zip 或 jt400.jar 必須在類別路徑中，才可呼叫此程式。
5. IBM Toolbox for Java 程式碼會丟出程式必須捕捉的異常。
6. AS400 類別可供 IBM Toolbox for Java 使用。此類別可管理登入資訊、建立及維護 socket 連線，以及傳送與接收資料。在此範例中，程式將傳送伺服器名稱給 AS400 物件。
7. VJobList 類別可供 IBM Toolbox for Java 使用，以呈現可在 Vaccess (GUI) 元件中顯示的伺服器工作清單。請注意，使用 AS400 物件可指定清單所在的伺服器。
8. 這一行會建構用來顯示工作清單的頁框或頂層視窗。
9. ErrorDialogAdapter 是一種 IBM Toolbox for Java 圖形式使用者介面 (GUI) 元件，每當應用程式有錯誤事件發生時，就會建立此元件來自動顯示對話視窗。
10. 此行可建立 AS400ExplorerPane，這是一個圖形式使用者介面 (GUI)，代表伺服器資源中的物件階層。AS400ExplorerPane 會在 VJobList 的左邊根部顯示樹狀結構，而在右邊顯示資源的明細。如此僅將窗格起始設定爲預設狀態，而不載入 VJobList 的內容到窗格中。
11. 此行會新增您在步驟 9 中建立的錯誤處理程式，作爲 VJobList 圖形式使用者介面 (GUI) 元件上的接收程式。
12. 此行會將 JobList 的內容載入 ExplorerPane。必須明確呼叫此方法，才能與伺服器通訊或從伺服器載入資訊。如此可讓應用程式控制與伺服器通訊的時間。有了這個方法，您可以：
    - 將窗格新增到頁框之前先載入內容。載入所有資訊之前將不顯示頁框，如本範例。
    - 將窗格新增到頁框且顯示該頁框之後再載入內容。頁框會出現「等待游標」，且在載入後該資訊就會填入。
13. 此行會新增一個視窗接收程式，以便在訊框關閉時結束應用程式。
14. 此行將工作清單圖形式使用者介面 GUI 元件新增至控制訊框中心。
15. 此行可以呼叫顯示方法，讓使用者可以看到視窗。
16. IBM Toolbox for Java 異常情況都已經過翻譯，因此文字會以工作站的語言顯示。例如，此程式可在錯誤處理程序時，顯示異常文字。
17. IBM Toolbox for Java 可以建立緒，以執行 IBM Toolbox for Java 活動。若程式在結束時沒有執行 System.exit(0)，可能無法正常結束。例如，如果是從 Windows 95 DOS 提示執行程式，且未使用這一行，當程式完成時，將不會返回指令提示。

範例：程式設計祕訣

本節將列出 IBM Toolbox for Java 程式設計要訣之文件所提供的所有程式碼範例。

管理連線

- 範例：以 CommandCall 物件建立 iSeries 伺服器連線
- 範例：以 CommandCall 物件建立兩個 iSeries 伺服器連線
- 範例：以 AS/400 物件建立 CommandCall 與 IFSFileInputStream 物件
- 範例：使用 AS400ConnectionPool 預先連接 iSeries 伺服器
- 範例：使用 AS400ConnectionPool 預先連接 iSeries 伺服器上的特定服務程式，然後重覆使用連線

啓動與結束連線

- 範例：Java 程式如何預先連接到 iSeries 伺服器
- 範例：Java 程式如何切斷 iSeries 伺服器連線
- 範例：Java 程式如何以 disconnectService() 及 run() 切斷 iSeries 伺服器連線並重新連線
- 範例：Java 程式如何切斷 iSeries 伺服器連線而無法重新連線

異常情況

- 範例：使用異常情況

錯誤事件

- 範例：處理錯誤事件
- 範例：定義錯誤接收器
- 範例：使用自訂的處理程式來處理錯誤事件

追蹤

- 範例：使用追蹤
- 範例：使用 setTraceOn()
- 範例：使用元件追蹤

最佳化

- 範例：建立兩個 AS400 物件
- 範例：使用 AS400 物件來代表第二個伺服器

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：ToolboxME for iSeries

本節將列出 IBM ToolboxME for iSeries 文件所提供的所有程式碼範例。

- 第 328 頁的『ToolboxME for iSeries 範例：JdbcDemo.java』

- 『範例：使用 ToolboxME for iSeries、MIDP 及 JDBC』
- 第 645 頁的『範例：使用 ToolboxME for iSeries、MIDP 及 IBM Toolbox for Java』

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：使用 ToolboxME for iSeries、MIDP 及 JDBC

下列原始程式說明 ToolboxME for iSeries 應用程式可以使用行動資訊裝置設定檔 (MIDP) 及 JDBC 來離線存取資料庫及儲存資訊的一種方式。

此範例示範一家房屋仲介商如何能夠檢視目前銷售的房屋以及出價。仲介商使用 Tier0 裝置存取房屋資訊，此資訊儲存在 iSeries 伺服器資料庫中。

當下列程式碼範例是建置成工作程式時，它連接到基於此用途而建立的資料庫。

要建立原始程式碼的工作版本及取得原始程式來建立必要的資料庫及大量輸入資料，您必須下載此範例。您也可以複查建立和執行此程式範例的指示。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// ToolboxME for iSeries example. This program is an example MIDlet that shows how
// you might code a JdbcMe application for the MIDP profile. Refer to the
// startApp, pauseApp, destroyApp and commandAction methods to see how it handles
// each requested transition.
//
////////////////////////////////////////////////////////////////////////////

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.sql.*;
import javax.microedition.rms.*;

import com.ibm.as400.micro.*;


public class JdbcMidpBid extends MIDlet implements CommandListener
{
        private static int BID_PROPERTY = 0;
        private Display   display;

    private TextField urlText = new TextField("urltext",
                                          "jdbc:as400://mySystem;user=myUid;password=myPwd;",
                                          65,
                                          TextField.ANY);
    private TextField jdbcmeText = new TextField("jdbcmetext", "meserver=myMEServer", 40, TextField.ANY);
    private TextField jdbcmeTraceText = new TextField("jdbcmetracetext", "0", 10, TextField.ANY);
    private final static String GETBIDS = "No bids are available, select here to download bids";
```

```
private List      main = new List("JdbcMe Bid Demo", Choice.IMPLICIT);
private List      listings = null;
private Form      aboutBox;
private Form      bidForm;
private Form      settingsForm;
private int       bidRow = 0;
private String    bidTarget = null;
private String    bidTargetKey = null;
private TextField bidText = new TextField("bidtext", "", 10, TextField.NUMERIC);
private Form      errorForm = null;

private Command exitCommand   = new Command("Exit", Command.SCREEN, 0);
private Command backCommand   = new Command("Back", Command.SCREEN, 0);
private Command cancelCommand = new Command("Cancel", Command.SCREEN, 0);
private Command goCommand     = new Command("Go", Command.SCREEN, 1);
private Displayable   onErrorGoBackTo = null;

/*
 * Construct a new JdbcMidpBid.
 */
public JdbcMidpBid()
{
    display = Display.getDisplay(this);
}

/**
 * Show the main screen
 */
public void startApp()
{
    main.append("Show Bids", null);
    main.append("Get New Bids", null);
    main.append("Settings", null);
    main.append("About", null);
    main.addCommand(exitCommand);
    main.setCommandListener(this);

    display.setCurrent(main);
}

public void commandAction(Command c, Displayable s)
{
    // All exitCommand processing is the same.
    if (c == exitCommand)
    {
        destroyApp(false);
        notifyDestroyed();
        return;
    }
    if (s instanceof List)
    {
        List     current = (List)s;

        // An action occurred on the main page
        if (current == main)
        {
            int   idx = current.getSelectedIndex();
            switch (idx)
            {
            case 0:     // Show current bids
                showBids();
                break;
            case 1:     // Get New Bids
                getNewBids();
                break;
            case 2:     // Settings
                doSettings();
```

```
            break;
        case 3:     // About
            aboutBox();
            break;
        default :
            break;
        }
        return;
    } // current == main

    // An action occurred on the listings page
    if (current == listings)
    {
        if (c == backCommand)
        {
            display.setCurrent(main);
            return;
        }
        if (c == List.SELECT_COMMAND)
        {
            int idx = listings.getSelectedIndex();
            String stext = listings.getString(idx);
            if (stext.equals(GETBIDS))
            {
                getNewBids();
                return;
            }
            int   commaIdx = stext.indexOf(',');
            bidTargetKey = stext.substring(0, commaIdx);
            bidTarget = stext.substring(commaIdx+1) + "\n";
            // Also keep track of which offline result set row
            // This is. It happens to be the same as the index
            // in the list.
            bidRow = idx;

            bidOnProperty();
        }
    } // current == listings
    return;
} // instanceof List
if (s instanceof Form)
{
    Form     current = (Form)s;
    if (current == errorForm)
    {
        if (c == backCommand)
            display.setCurrent(onErrorGoBackTo);

        return;
    } // errorForm
    if (current == settingsForm)
    {
        if (c == backCommand)
        {
            // Done with settings.
            display.setCurrent(main);
            settingsForm = null;
            return;
        }
    } // settingsForm
    if (current == aboutBox)
    {
        if (c == backCommand)
        {
            // Done with about box.
            display.setCurrent(main);
            aboutBox = null;
```

```
                return;
            }
        }
        if (current == bidForm)
        {
            if (c == cancelCommand)
            {
                display.setCurrent(listings);
                bidForm = null;
                return;
            }
            if (c == goCommand)
            {
                submitBid();
                if (display.getCurrent() != bidForm)
                {
                    // If we're no longer positioned at the
                    // bidForm, we will get rid of it.
                    bidForm = null;
                }
                return;
            }
            return;
        } // current == bidForm
    } // instanceof Form
}

public void aboutBox()
{
    aboutBox = new Form("aboutbox");
    aboutBox.setTitle("About");
    aboutBox.append(new StringItem("", "Midp RealEstate example for JdbcMe "));
    aboutBox.addCommand(backCommand);
    aboutBox.setCommandListener(this);
    display.setCurrent(aboutBox);
}

/**
 *  The settings form.
 */
public void doSettings()
{
    settingsForm = new Form("settingsform");
    settingsForm.setTitle("Settings");
    settingsForm.append(new StringItem("", "DB URL"));
    settingsForm.append(urlText);
    settingsForm.append(new StringItem("", "JdbcMe server"));
    settingsForm.append(jdbcmeText);
    settingsForm.append(new StringItem("", "Trace"));

    settingsForm.addCommand(backCommand);
    settingsForm.setCommandListener(this);
    display.setCurrent(settingsForm);
}


/**
 * Show the bid screen for the bid target
 * that we selected.
 */
public void bidOnProperty()
{
    StringItem item = new StringItem("", bidTarget);

    bidText = new TextField("bidtext", "", 10, TextField.NUMERIC);
    bidText.setString("");
```

```
    bidForm = new Form("bidform");
    bidForm.setTitle("Submit a bid for:");
    BID_PROPERTY = 0;
    bidForm.append(item);
    bidForm.append(new StringItem("", "Your bid:"));
    bidForm.append(bidText);
    bidForm.addCommand(cancelCommand);
    bidForm.addCommand(goCommand);
    bidForm.setCommandListener(this);
    display.setCurrent(bidForm);
}

/**
 * Update the listings card with the
 * current list of bids that we are interested in.
 */
public void getNewBids()
{
    // Reset the old listing
    listings = null;
    listings = new List("JdbcMe Bids", Choice.IMPLICIT);
    java.sql.Connection     conn = null;
    Statement               stmt = null;
    try
    {
        conn = DriverManager.getConnection(urlText.getString() + ";" + jdbcmeText.getString());

        stmt = conn.createStatement();

        // Since we do not want the prepared statement to persist,
        // a normal statement is really better in this environemnt.
        String sql = "select mls, address, currentbid from qjdbcme.realestate where currentbid <> 0";

        boolean results =((JdbcMeStatement)stmt).executeToOfflineData(sql,
                                                                      "JdbcMidpBidListings",
                                                                      0,
                                                                      0);
        if (results)
        {
            setupListingsFromOfflineData();
        }
        else
        {
            listings.append("No bids found", null);
            listings.addCommand(backCommand);
            listings.setCommandListener(this);
        }
    }
    catch (Exception e)
    {
        // Currently no valid listings retrieved, so lets
        // reset it to empty.
        listings = new List("JdbcMe Bids", Choice.IMPLICIT);
        listings.append(GETBIDS, null);
        listings.addCommand(backCommand);
        listings.setCommandListener(this);

        // Return to main after showing the error.
        showError(main, e);
        return;
    }
    finally
    {
        if (conn != null)
        {
            try
            {
```

```
                conn.close();
            }
            catch (Exception e)
            {
            }
        }
        conn = null;
        stmt = null;
    }
    showBids();
}

public void setupListingsFromOfflineData()
{
    // Skip the first four rows in the record store
    // (eyecatcher, version, num columns, sql column
    // types)
    // and each subsequent row in the record store is
    // a single column. Our query returns 3 columns which
    // we will return concatenated as a single string.
    ResultSet         rs = null;
    listings.addCommand(backCommand);
    listings.setCommandListener(this);
    try
    {
        int                     i = 5;
        int                     max = 0;
        StringBuffer   buf  = new StringBuffer(20);

        // Creator and dbtype unused in MIDP
        rs = new JdbcMeOfflineResultSet("JdbcMidpBidListings", 0, 0);
        if (rs == null)
        {
            // New listings...
            listings = new List("JdbcMe Bids", Choice.IMPLICIT);
            listings.append(GETBIDS, null);
            listings.addCommand(backCommand);
            listings.setCommandListener(this);
            return;
        }

        i = 0;
        String      s = null;
        while (rs.next())
        {
            ++i;

            s = rs.getString(1);
            buf.append(s);

            buf.append(",");
            s = rs.getString(2);
            buf.append(s);

            buf.append(", $");
            s = rs.getString(3);
            buf.append(s);

            listings.append(buf.toString(), null);
            buf.setLength(0);
        }

        if (i == 0)
        {
            listings.append("No bids found", null);
            return;
        }
```

```
    }
    catch (Exception e)
    {
        // Currently no valid listings retrieved, so lets
        // reset it to empty.
        listings = new List("JdbcMe Bids", Choice.IMPLICIT);
        listings.append(GETBIDS, null);
        listings.addCommand(backCommand);
        listings.setCommandListener(this);

        // Return to main after showing the error.
        showError(main, e);
        return;
    }
    finally
    {
        if (rs != null)
        {
            try
            {
                rs.close();
            }
            catch (Exception e)
            {
            }
            rs = null;
        }
        System.gc();
    }
}

/**
 * Update the listings card with the
 * current list of bids that we are interested in.
 */
public void submitBid()
{
    java.sql.Connection     conn = null;
    Statement               stmt = null;
    try
    {
        conn = DriverManager.getConnection(urlText.getString() + ";" + jdbcmeText.getString());

        stmt = conn.createStatement();

        // Since we do not want the prepared statement to persist,
        // a normal statement is really better in this environemnt.
        StringBuffer   buf = new StringBuffer(100);
        buf.append("Update QJdbcMe.RealEstate Set CurrentBid = ");
        buf.append(bidText.getString());
        buf.append(" Where MLS = '");
        buf.append(bidTargetKey);
        buf.append("' and CurrentBid < ");
        buf.append(bidText.getString());
        String      sql = buf.toString();

        int   updated = stmt.executeUpdate(sql);
        if (updated == 1)
        {
            // BID Accepted.
            String oldS = listings.getString(bidRow);
            int   commaIdx = bidTarget.indexOf(',');
            String bidAddr = bidTarget.substring(0, commaIdx);

            String newS = bidTargetKey + "," + bidAddr + ", $" + bidText.getString();

            ResultSet         rs = null;
```

```
            try
            {
                // Creator and dbtype unused in MIDP
                rs = new JdbcMeOfflineResultSet("JdbcMidpBidListings", 0, 0);
                rs.absolute(bidRow+1);
                rs.updateString(3, bidText.getString());
                rs.close();
            }
            catch (Exception e)
            {
                if (rs != null)
                    rs.close();
            }

            // Also update our live list of that result set.
            listings.set(bidRow, newS, null);
            display.setCurrent(listings);
            conn.commit();
        }
        else
        {
            conn.rollback();
            throw new SQLException("Failed to bid, someone beat you to it");
        }
    }
    catch (SQLException e)
    {
        // Return to the bid form after showing the error.
        showError(bidForm, e);
        return;
    }
    finally
    {
        if (conn != null)
        {
            try
            {
                conn.close();
            }
            catch (Exception e)
            {
            }
        }
        conn = null;
        stmt = null;
    }

    // Exit without exception, then show the current bids
    showBids();
}

/**
 * Show an error condition.
 */
public void showError(Displayable d, Exception e)
{
    String s = e.toString();

    onErrorGoBackTo = d;
    errorForm = new Form("Error");
    errorForm.setTitle("SQL Error");
    errorForm.append(new StringItem("", s));
    errorForm.addCommand(backCommand);
    errorForm.setCommandListener(this);
    display.setCurrent(errorForm);
}
```

```
    /**
     * Show the current bids.
     */
    public void showBids()
    {
        if (listings == null)
        {
            // If we have no current listings, lets set
            // them up.
            listings = new List("JdbcMe Bids", Choice.IMPLICIT);
            setupListingsFromOfflineData();
        }
        display.setCurrent(listings);
    }

    /**
     * Time to pause, free any space we do not need right now.
     */
    public void pauseApp()
    {
        display.setCurrent(null);
    }

    /**
     * Destroy must cleanup everything.
     */
    public void destroyApp(boolean unconditional)
    {
    }
}
```

範例：使用 ToolboxME for iSeries、MIDP 及 IBM Toolbox for Java

下列原始程式說明 ToolboxME for iSeries 應用程式可以使用行動資訊裝置設定檔 (MIDP) 及 IBM Toolbox for Java 來存取 iSeries 伺服器上之資料及服務的一種方式。

此範例將示範內建於 IBM Toolbox for Java 2 Micro Edition 支援中的每一個功能。此應用程式的特性是一些不一樣的頁面或畫面，說明 Tier0 裝置可使用這些功能的一些方式。

當下列程式碼範例建置爲工作程式時，會使用「程式呼叫標記語言 (PCML)」檔案在 iSeries 伺服器上執行指令。

要建立原始程式碼的工作版本及取得必要的 PCML 來源來執行伺服器指令，您必須下載此範例。您也可以複查建立和執行此程式範例的指示。

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////////
//
// ToolboxME for iSeries example. This program is an example that shows how
// ToolboxME for iSeries can use PCML to access data and services on an
// iSeries server.
//
// This application requires that the qsyrusri.pcml file is present in the
// CLASSPATH of the MEServer.
//
///////////////////////////////////////////////////////////////////////////////

import java.io.*;
import java.sql.*;
import java.util.Hashtable;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.rms.*;
```

```
import com.ibm.as400.micro.*;


public class ToolboxMidpDemo extends MIDlet implements CommandListener
{
        private Display   display_;

        // A ToolboxME system object.
    private AS400 system_;

    private List      main_ = new List("ToolboxME MIDP Demo", Choice.IMPLICIT);

        // Create a form for each component.
        private Form      signonForm_;
        private Form      cmdcallForm_;
        private Form      pgmcallForm_;
        private Form      dataqueueForm_;
        private Form      aboutForm_;

        // Visable Text for each component.
    static final String SIGN_ON       = "SignOn";
    static final String COMMAND_CALL  = "CommandCall";
    static final String PROGRAM_CALL  = "ProgramCall";
    static final String DATA_QUEUE    = "DataQueue";
    static final String ABOUT         = "About";

    static final String NOT_SIGNED_ON = "Not signed on.";
    static final String DQ_READ       = "Read";
    static final String DQ_WRITE      = "Write";

        // A ticker to display the signon status.
        private Ticker    ticker_ = new Ticker(NOT_SIGNED_ON);

        // Commands that can be performed.
    private static final Command actionExit_   = new Command("Exit", Command.SCREEN, 0);
    private static final Command actionBack_   = new Command("Back", Command.SCREEN, 0);
    private static final Command actionGo_     = new Command("Go", Command.SCREEN, 1);
    private static final Command actionClear_  = new Command("Clear", Command.SCREEN, 1);
    private static final Command actionRun_    = new Command("Run", Command.SCREEN, 1);
        private static final Command actionSignon_ = new Command(SIGN_ON, Command.SCREEN, 1);
    private static final Command actionSignoff_= new Command("SignOff", Command.SCREEN, 1);

    private Displayable   onErrorGoBackTo_;  // the form to return to when done displaying the error form

        // TextFields for the SignOn form.
    private TextField signonSystemText_ = new TextField("System", "rchasdm3", 20, TextField.ANY);
    private TextField signonUidText_ = new TextField("UserId", "JAVA", 10, TextField.ANY);
    // TBD temporary
    private TextField signonPwdText_ = new TextField("Password", "JTEAM1", 10, TextField.PASSWORD);
    private TextField signonServerText_ = new TextField("MEServer", "localhost", 10, TextField.ANY);
    private StringItem signonStatusText_ = new StringItem("Status", NOT_SIGNED_ON);

        // TextFields for the CommandCall form.
    // TBD: max size; TBD: TextBox???
    private TextField cmdText_ = new TextField("Command", "CRTLIB FRED", 256, TextField.ANY);
    private StringItem cmdMsgText_ = new StringItem("Messages", null);
    private StringItem cmdStatusText_ = new StringItem("Status", null);

        // TextFields for the ProgramCall form.
    private StringItem pgmMsgDescription_ = new StringItem("Messages", null);
    private StringItem pgmMsgText_ = new StringItem("Messages", null);

        // TextFields for the DataQueue form.
    private TextField dqInputText_ = new TextField("Data to write", "Hi there", 30, TextField.ANY);
    private StringItem dqOutputText_ = new StringItem("DQ contents", null);
    private ChoiceGroup dqReadOrWrite_ = new ChoiceGroup("Action",
```

```
                                                    Choice.EXCLUSIVE,
                                                    new String[] { DQ_WRITE, DQ_READ},
                                                    null);
private StringItem dqStatusText_ = new StringItem("Status", null);


/**
 * Creates a new ToolboxMidpDemo.
 **/
public ToolboxMidpDemo()
{
    display_ = Display.getDisplay(this);
    // Note: The KVM-based demo used TabbedPane for the main panel.
    // MIDP has no similar class, so we use a List instead.
}

/**
 * Show the main screen.
 * Implements abstract method of class Midlet.
 **/
protected void startApp()
{
    main_.append(SIGN_ON, null);
    main_.append(COMMAND_CALL, null);
    main_.append(PROGRAM_CALL, null);
    main_.append(DATA_QUEUE, null);
    main_.append(ABOUT, null);

    main_.addCommand(actionExit_);
    main_.setCommandListener(this);

    display_.setCurrent(main_);
}

// Implements method of interface CommandListener.
public void commandAction(Command action, Displayable dsp)
{
    // All 'exit' and 'back' processing is the same.
    if (action == actionExit_)
    {
        destroyApp(false);

        notifyDestroyed();
    }
    else if (action == actionBack_)
    {
        // Return to main menu.
        display_.setCurrent(main_);
    }
    else if (dsp instanceof List)
    {
        List current = (List)dsp;

        // An action occurred on the main page
        if (current == main_)
        {
            int   idx = current.getSelectedIndex();

            switch (idx)
            {
            case 0:     // SignOn
                showSignonForm();
                break;
            case 1:     // CommandCall
                showCmdForm();
                break;
            case 2:     // ProgramCall
```

```
            showPgmForm();
            break;
        case 3:     // DataQueue
            showDqForm();
            break;
        case 4:     // About
            showAboutForm();
            break;
        default:    // None of the above
            feedback("Internal error: Unhandled selected index in main: " + idx,
                     AlertType.ERROR);
            break;
        }
    } // current == main
    else
        feedback("Internal error: The Displayable object is a List but is not main_.",
                 AlertType.ERROR);
} // instanceof List
else if (dsp instanceof Form)
{
    Form current = (Form)dsp;

    if (current == signonForm_)
    {
        if (action == actionSignon_)
        {
            // Create a ToolboxME system object.
            system_ = new AS400(signonSystemText_.getString(),
                                signonUidText_.getString(),
                                signonPwdText_.getString(),
                                signonServerText_.getString());

            try
            {
                // Connect to the iSeries.
                system_.connect();

                // Set the signon status text.
                signonStatusText_.setText("Signed on.");

                // Display a confirmation dialog that the user is signed on.
                feedback("Successfully signed on.", AlertType.INFO, main_);

                // Replace the SignOn button with SignOff.
                signonForm_.removeCommand(actionSignon_);
                signonForm_.addCommand(actionSignoff_);

                // Update the ticker.
                ticker_.setString("... Signed on to '" +
                                  signonSystemText_.getString() + "' as '" +
                                  signonUidText_.getString() + "' via '" +
                                  signonServerText_.getString() + "' ... ");
            }
            catch (Exception e)
            {
                e.printStackTrace();

                // Set the signon status text.
                signonStatusText_.setText(NOT_SIGNED_ON);

                feedback("Signon failed.  " + e.getMessage(), AlertType.ERROR);
            }
        }
        else if (action == actionSignoff_)
        {
            if (system_ == null)
                feedback("Internal error: System is null.", AlertType.ERROR);
```

```
        else
        {
            try
            {
                // Disconnect from the iSeries.
                system_.disconnect();
                system_ = null;

                // Set the signon status text.
                signonStatusText_.setText(NOT_SIGNED_ON);

                // Display a confirmation dialog that the user is no longer signed on.
                feedback("Successfully signed off.", AlertType.INFO, main_);

                // Replace the SignOff button with SignOn.
                signonForm_.removeCommand(actionSignoff_);
                signonForm_.addCommand(actionSignon_);

                // Update the ticker.
                ticker_.setString(NOT_SIGNED_ON);
            }
            catch (Exception e)
            {
                feedback(e.toString(), AlertType.ERROR);

                e.printStackTrace();

                signonStatusText_.setText("Error.");

                feedback("Error during signoff.", AlertType.ERROR);
            }
        }
    }
    else  // None of the above.
    {
        feedback("Internal error: Action is not recognized.", AlertType.INFO);
    }
} // signonForm_
else if (current == cmdcallForm_)
{
    if (action == actionRun_)
    {
        // If the user has not signed on, display an alert.
        if (system_ == null)
        {
            feedback(NOT_SIGNED_ON, AlertType.ERROR);
            return;
        }

        // Get the command the user entered in the wireless device.
        String cmdString = cmdText_.getString();

        // If the command was not specified, display an alert.
        if (cmdString == null || cmdString.length() == 0)
            feedback("Specify command.", AlertType.ERROR);
        else
        {
            try
            {
                // Run the command.
                String[] messages = CommandCall.run(system_, cmdString);

                StringBuffer status = new StringBuffer("Command completed with ");

                // Check to see if their are any messages.
                if (messages.length == 0)
                {
```

```
                    status.append("no returned messages.");

                    cmdMsgText_.setText(null);

                    cmdStatusText_.setText("Command completed successfully.");
                }
                else
                {
                    if (messages.length == 1)
                        status.append("1 returned message.");
                    else
                        status.append(messages.length + " returned messages.");

                    // If there are messages, display only the first message.
                    cmdMsgText_.setText(messages[0]);

                    cmdStatusText_.setText(status.toString());
                }

                repaint();
            }
            catch (Exception e)
            {
                feedback(e.toString(), AlertType.ERROR);

                e.printStackTrace();

                feedback("Error when running command.", AlertType.ERROR);
            }
        }
    }
    else if (action == actionClear_)
    {
        // Clear the command text and messages.
        cmdText_.setString("");

        cmdMsgText_.setText(null);

        cmdStatusText_.setText(null);

        repaint();
    }
    else  // None of the above.
    {
        feedback("Internal error: Action is not recognized.", AlertType.INFO);
    }
} // cmdcallForm_
else if (current == pgmcallForm_)
{
    if (action == actionRun_)
    {
        // If the user is not signed on before doing a program call, display an alert.
        if (system_ == null)
        {
            feedback(NOT_SIGNED_ON, AlertType.ERROR);
            return;
        }

        pgmMsgText_.setText(null);

        // See the PCML example in the IBM Toolbox for Java information.
        String pcmlName = "qsyrusri.pcml"; // The PCML file we want to use.
        String apiName = "qsyrusri";

        // Create a hashtable that contains the input parameters for the program call.
        Hashtable parmsToSet = new Hashtable(2);
        parmsToSet.put("qsyrusri.receiverLength", "2048");
```

```
    parmsToSet.put("qsyrusri.profileName", signonUidText_.getString().toUpperCase());

    // Create a string array that contains the output parameters to retrieve.
    String[] parmsToGet = { "qsyrusri.receiver.userProfile",
                            "qsyrusri.receiver.previousSignonDate",
                            "qsyrusri.receiver.previousSignonTime",
                            "qsyrusri.receiver.daysUntilPasswordExpires"};

    // A string array containing the descriptions of the parameters to display.
    String[] displayParm = { "Profile",
                             "Last signon Date",
                             "Last signon Time",
                             "Password Expired (days)"};

    try
    {
        // Run the program.
        String[] valuesToGet = ProgramCall.run(system_,
                                               pcmlName,
                                               apiName,
                                               parmsToSet,
                                               parmsToGet);

        // Create a StringBuffer and add each of the parameters we retreived.
                                StringBuffer txt = new StringBuffer();
        txt.append(displayParm[0] + ": " + valuesToGet[0] + "\n");

        char[] c = valuesToGet[1].toCharArray();
        txt.append(displayParm[1] + ": " + c[3] + c[4] + "/" +
                   c[5] + c[6] + "/" + c[1] + c[2] + "\n");

        char[] d = valuesToGet[2].toCharArray();
        txt.append(displayParm[2] + ": " + d[0] + d[1] + ":" + d[2] + d[3] + "\n");
        txt.append(displayParm[3] + ": " + valuesToGet[3] + "\n");

        // Set the displayable text of the program call results.
        pgmMsgText_.setText(txt.toString());

        StringBuffer status = new StringBuffer("Program completed with ");

        if (valuesToGet.length == 0)
        {
            status.append("no returned values.");

            feedback(status.toString(), AlertType.INFO);
        }
        else
        {
            if (valuesToGet.length == 1)
                status.append("1 returned value.");
            else
                status.append(valuesToGet.length + " returned values.");

            feedback(status.toString(), AlertType.INFO);
        }
    }
    catch (Exception e)
    {
        feedback(e.toString(), AlertType.ERROR);

        e.printStackTrace();

        feedback("Error when running program.", AlertType.ERROR);
    }
}
else if (action == actionClear_)
{
```

```
            // Clear the program call results.
            pgmMsgText_.setText(null);

            repaint();
        }
    }  // pgmcallForm_
    else if (current == dataqueueForm_)  // DataQueue
    {
        if (action == actionGo_)
        {
            // If the user has not signed on before performing Data Queue actions,
            // display an alert.
            if (system_ == null)
            {
                feedback(NOT_SIGNED_ON, AlertType.ERROR);

                return;
            }

            // Create a library to create the data queue in.
            try
            {
                CommandCall.run(system_, "CRTLIB FRED");
            }
            catch (Exception e)
            {
            }

            // Run a command to create a data queue.
            try
            {
                CommandCall.run(system_, "CRTDTAQ FRED/MYDTAQ MAXLEN(2000)");
            }
            catch (Exception e)
            {
                feedback("Error when creating data queue.  " + e.getMessage(),
                         AlertType.WARNING);
            }

            try
            {
                // See which action was selected (Read or Write).
                if (dqReadOrWrite_.getString(dqReadOrWrite_.getSelectedIndex()).equals(DQ_WRITE))
                {
                    // Write
                    dqOutputText_.setText(null);

                    // Get the text from the wireless device input to be written to
                    // the data queue.
                    if (dqInputText_.getString().length() == 0)
                        dqStatusText_.setText("No data specified.");
                    else
                    {
                        // Write to the data queue.
                        DataQueue.write(system_,
                                        "/QSYS.LIB/FRED.LIB/MYDTAQ.DTAQ",
                                        dqInputText_.getString().getBytes() );

                        dqInputText_.setString(null);

                        // Display the status.
                        dqStatusText_.setText("The 'write' operation completed.");
                    }
                }
                else  // Read
                {
                    // Read from the data queue.
```

```
                        byte[] b = DataQueue.readBytes(system_, "/QSYS.LIB/FRED.LIB/MYDTAQ.DTAQ");

                        // Determine if the data queue contained entries or not
                        // and display the appropriate message.
                        if (b == null)
                        {
                            dqStatusText_.setText("No dataqueue entries are available.");

                            dqOutputText_.setText(null);
                        }
                        else if (b.length == 0)
                        {
                            dqStatusText_.setText("Dataqueue entry has no data.");

                            dqOutputText_.setText(null);
                        }
                        else
                        {
                            dqStatusText_.setText("The 'read' operation completed.");

                            dqOutputText_.setText(new String(b));
                        }
                    }

                    repaint();
                }
                catch (Exception e)
                {
                    e.printStackTrace();

                    feedback(e.toString(), AlertType.ERROR);

                    feedback("Error when running command.  " + e.getMessage(), AlertType.ERROR);
                }
            }  // actionGo_
            else if (action == actionClear_)
            {
                // Clear the data queue form.
                dqInputText_.setString("");

                dqOutputText_.setText(null);

                dqReadOrWrite_.setSelectedFlags(new boolean[] { true, false});

                dqStatusText_.setText(null);

                repaint();
            }
            else  // None of the above.
            {
                feedback("Internal error: Action is not recognized.", AlertType.INFO);
            }
        } // dataqueueForm_
        else if (current == aboutForm_)  // "About".
        {
            // Should never reach here, since the only button is "Back".
        }  // None of the above.
        else
            feedback("Internal error: Form is not recognized.", AlertType.ERROR);
    } // instanceof Form
    else
        feedback("Internal error: Displayable object not recognized.", AlertType.ERROR);
}



/**
```

```
     *  Displays the "About" form.
     **/
    private void showAboutForm()
    {
        // If the about form is null, create and append it.
        if (aboutForm_ == null)
        {
            aboutForm_ = new Form(ABOUT);
            aboutForm_.append(new StringItem(null,
                              "This is a MIDP example application that uses the " +
                              "IBM Toolbox for Java Micro Edition (ToolboxME)."));

            aboutForm_.addCommand(actionBack_);
            aboutForm_.setCommandListener(this);
        }

        display_.setCurrent(aboutForm_);
    }



    /**
     *  Displays the "SignOn" form.
     **/
    private void showSignonForm()
    {
        // Create the signon form.
        if (signonForm_ == null)
        {
            signonForm_ = new Form(SIGN_ON);
            signonForm_.append(signonSystemText_);
            signonForm_.append(signonUidText_);
            signonForm_.append(signonPwdText_);
            signonForm_.append(signonServerText_);
            signonForm_.append(signonStatusText_);
            signonForm_.addCommand(actionBack_);
            signonForm_.addCommand(actionSignon_);
            signonForm_.setCommandListener(this);
            signonForm_.setTicker(ticker_);
        }

        display_.setCurrent(signonForm_);
    }



    /**
     *  Displays the "CommandCall" form.
     **/
    private void showCmdForm()
    {
        // Create the command call form.
        if (cmdcallForm_ == null)
        {
            cmdcallForm_ = new Form(COMMAND_CALL);
            cmdcallForm_.append(cmdText_);
            cmdcallForm_.append(cmdMsgText_);
            cmdcallForm_.append(cmdStatusText_);
            cmdcallForm_.addCommand(actionBack_);
            cmdcallForm_.addCommand(actionClear_);
            cmdcallForm_.addCommand(actionRun_);
            cmdcallForm_.setCommandListener(this);
            cmdcallForm_.setTicker(ticker_);
        }

        display_.setCurrent(cmdcallForm_);
    }
```

```
/**
 *  Displays the "ProgramCall" form.
 **/
private void showPgmForm()
{
    // Create the program call form.
    if (pgmcallForm_ == null)
    {
        pgmcallForm_ = new Form(PROGRAM_CALL);
        pgmcallForm_.append(new StringItem(null,
                                           "This calls the Retrieve User Information (QSYRUSRI) " +
                                           "API, and returns information about the current " +
                                           "user profile."));
        pgmcallForm_.append(pgmMsgText_);
        pgmcallForm_.addCommand(actionBack_);
        pgmcallForm_.addCommand(actionClear_);
        pgmcallForm_.addCommand(actionRun_);
        pgmcallForm_.setCommandListener(this);
        pgmcallForm_.setTicker(ticker_);
    }

    display_.setCurrent(pgmcallForm_);
}


/**
 *  Displays the "DataQueue" form.
 **/
private void showDqForm()
{
    // Create the data queue form.
    if (dataqueueForm_ == null)
    {
        dataqueueForm_ = new Form(DATA_QUEUE);
        dataqueueForm_.append(dqInputText_);
        dataqueueForm_.append(dqOutputText_);
        dataqueueForm_.append(dqReadOrWrite_);
        dataqueueForm_.append(dqStatusText_);
        dataqueueForm_.addCommand(actionBack_);
        dataqueueForm_.addCommand(actionClear_);
        dataqueueForm_.addCommand(actionGo_);
        dataqueueForm_.setCommandListener(this);
        dataqueueForm_.setTicker(ticker_);
    }

    display_.setCurrent(dataqueueForm_);
}


private void feedback(String text, AlertType type)
{
    feedback(text, type, display_.getCurrent());
}

/**
 *  This method is used to create a dialog and display feedback
 *  information using an Alert to the user.
 **/
private void feedback(String text, AlertType type, Displayable returnToForm)
{
    System.err.flush();
    System.out.flush();

    Alert alert = new Alert("Alert", text, null, type);
```

```
        if (type == AlertType.INFO)
            alert.setTimeout(3000);  // milliseconds
        else
            alert.setTimeout(Alert.FOREVER);  // Require user to dismiss the alert.

        display_.setCurrent(alert, returnToForm);
    }


    // Force a repaint of the current form.
    private void repaint()
    {
        Alert alert = new Alert("Updating display ...", null, null, AlertType.INFO);
        alert.setTimeout(1000);  // milliseconds

        display_.setCurrent(alert, display_.getCurrent());
    }



    /**
     * Time to pause, free any space we don't need right now.
     * Implements abstract method of class Midlet.
     **/
    protected void pauseApp()
    {
        display_.setCurrent(null);
    }



    /**
     * Destroy must cleanup everything.
     * Implements abstract method of class Midlet.
     **/
    protected void destroyApp(boolean unconditional)
    {
        // Disconnect from the iSeries if the Midlet is being destroyed or exited.
        if (system_ != null)
        {
            try
            {
                system_.disconnect();
            }
            catch (Exception e)
            {
            }
        }
    }
}
```

範例：Utility 類別

本節將列出 IBM Toolbox for Java Utility 類別之文件所提供的所有程式碼範例。

AS/400ToolboxJarMaker

- 範例：從 jt400.jar 取出 AS400.class 及其所有相依類別
- 範例：將 jt400.jar 分成一組 300KB 的檔案
- 範例：從 JAR 檔案除去未使用的檔案
- 範例：使用 -ccsid 參數略過轉換表，藉以建立較小的 400KB JAR 檔

CommandPrompter

- 範例：使用 CommandPrompter 進行提示及執行指令

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：使用 CommandPrompter

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////////////
//
// CommandPrompter example. This program uses CommandPrompter, CommandCall, and
// AS400Message to prompt for a command, run the command, and display any
// messages returned if the command does not run.
//
// Command syntax:
//    Prompter commandString
//
////////////////////////////////////////////////////////////////////////////////

import com.ibm.as400.ui.util.CommandPrompter;
import com.ibm.as400.access.AS400;
import com.ibm.as400.access.AS400Message;
import com.ibm.as400.access.CommandCall;
import javax.swing.JFrame;
import java.awt.FlowLayout;
public class Prompter
{
public static void main ( String args[] ) throws Exception
  {
    JFrame frame = new JFrame();
    frame.getContentPane().setLayout(new FlowLayout());
  AS400 system = new AS400("mySystem", "myUserId", "myPasswd");
    String cmdName = args[0];

    // Launch the CommandPrompter
    CommandPrompter cp = new CommandPrompter(frame, system, cmdName);
    if (cp.showDialog() == CommandPrompter.OK)
    {
        String cmdString = cp.getCommandString();
    System.out.println("Command string: " + cmdString);

        // Run the command that was built in the prompter.
        CommandCall cmd = new CommandCall(system, cmdString);
        if (!cmd.run())
      {
            AS400Message[] msgList = cmd.getMessageList();
            for (int i = 0; i < msgList.length; ++i)
        {
           System.out.println(msgList[i].getText());
        }
      }
```

```
    }
        System.exit(0);
  }
}
```

範例：Vaccess 類別

本節會列出 IBM Toolbox for Java Vaccess 類別之文件所提供的所有程式碼範例。

AS400Panes

- 範例：建立 AS400DetailsPane 來呈現 systemAS400DetailsPane 上使用者定義的清單
- 範例：下面範例會在將明細窗格的內容新增到訊框之前先載入該內容
- 範例：使用 AS400ListPane 來呈現使用者清單
- 範例：使用 AS400DetailsPane 來顯示從指令呼叫傳回的訊息
- 範例：使用 AS400TreePane 來顯示目錄的樹狀檢視畫面
- 範例：使用 AS400ExplorerPane 來呈現不同的列印資源

指令呼叫

- 範例：建立 CommandCallButton
- 範例：新增 ActionCompletedListener 來處理指令所產生的所有 iSeries 訊息
- 範例：使用 CommandCallMenuItem

資料佇列

- 範例：建立 DataQueueDocument
- 範例：使用 DataQueueDocument

錯誤事件

- 範例：處理錯誤事件
- 範例：定義錯誤接收器
- 範例：使用自訂的處理程式來處理錯誤事件

整合檔案系統

- 範例：使用 IFSFileDialog
- 範例：使用 IFSFileSystemView
- 範例：使用 IFSTextFileDocument

JDBC

- 範例：使用 JDBC 驅動程式來建立及擴大表格。
- 範例：使用 JDBC 驅動程式來查詢表格並輸出它的內容。
- 範例：建立 AS400JDBCDataSourcePane

工作

- 範例：建立 VJobList，並以 AS400ExplorerPane 呈現清單
- 範例：在探索器窗格中呈現工作的清單

訊息

- 範例：使用 VMessageQueue

程式呼叫

- 範例：建立 ProgramCallMenuItem
- 範例：處理所有程式產生的 iSeries 訊息
- 範例：新增兩個參數
- 範例：在應用程式中使用 ProgramCallButton

列印

- 範例：使用 VPrinter
- 範例：VPrinterOutput

記錄層次存取

- 範例：建立一個 RecordListTablePane 物件，來顯示小於或等於 某個索引的所有記錄
- 範例：使用 RecordListFormPane

SpooledFileViewer

- 範例：建立排存檔檢視器來檢視先前在 iSeries 上建立的排存檔

SQL

- 範例：使用 SQLQueryBuilderPane
- 範例：使用 SQLResultSetTablePane

系統值

- 範例：使用 AS400Explorer 窗格建立一個系統值 GUI

使用者和群組

- 範例：透過 AS400DetailsPane 建立 VUserList
- 範例：使用 AS400ListPane，建立可供選擇的使用者清單

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：使用 VUserList

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// VUserList example.This program presents a list of users on
// a system in a list pane, and allows selection of one or more
// users.
//
```

```
// Command syntax:
//VUserListExample system
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VUserListExample
{


    private static AS400ListPane listPane;


    public static void main (String[] args)
    {
        // If a system is not specified, display help text and
        // exit.
        if (args.length != 1)
        {
            System.out.println("Usage:  VUserListExample system");
            return;
        }

        try
        {
            // Create an AS400 object.  The system name is passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);

            // Create a VUserList.This represents a list of users
// displayed in the list pane.
            VUserList userList = new VUserList (system);

            // Create a frame.
            JFrame f = new JFrame ("VUserList example");

            // Create an error dialog adapter.This displays
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

            // Create a list pane to display the user list.
            // Use load to get the information from the server.
            listPane = new AS400ListPane (userList);
            listPane.addErrorListener (errorHandler);

listPane.load ();

            // When the frame closes, report the selected
            // users and exit.
            f.addWindowListener (new WindowAdapter () {
                public void windowClosing (WindowEvent event)
                {
                    reportSelectedUsers ();
                    System.exit (0);
                }
            });

            // Layout the frame with the list pane.
            f.getContentPane ().setLayout (new BorderLayout ());
            f.getContentPane ().add ("Center", listPane);
            f.pack ();
            f.show ();
```

```
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
           System.exit (0);
        }
    }


    private static void reportSelectedUsers ()
    {
        VObject[] selectedUsers = listPane.getSelectedObjects ();

        if (selectedUsers.length == 0)
            System.out.println ("No users were selected.");
        else
        {
            System.out.println ("The selected users were:");
            for (int i = 0; i < selectedUsers.length; ++i)
                System.out.println (selectedUsers[i]);
        }
    }


}
```

範例：使用 VMessageList

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// VMessageList example.This program presents a details
// view of messages returned from a command call.
//
// Command syntax:
//VMessageListExample system
//
// This source is an example of IBM Toolbox for Java "VMessageList".
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VMessageListExample
{


    public static void main (String[] args)
    {
        // If a system was not specified, then display help text and
        // exit.
        if (args.length != 1)
        {
            System.out.println("Usage:  VMessageListExample system");
            return;
        }

        try
        {
            // Create an AS400 object.  The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);
```

```
            // Create a CommandCall object a run the command.
            CommandCall command = new CommandCall (system);
            command.run ("CRTLIB FRED");

            // Create a VMessageList object with the messages
            // returned from the command call.
            VMessageList messageList = new VMessageList (command.getMessageList ());

            // Create a frame.
            JFrame f = new JFrame ("VMessageList example");

            // Create an error dialog adapter.This will display
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

            // Create a details pane to display the message list.
            // Use load to load the information.
            AS400DetailsPane detailsPane = new AS400DetailsPane (messageList);
            detailsPane.addErrorListener (errorHandler);

detailsPane.load ();

            // When the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
                public void windowClosing (WindowEvent event)
                {
                    System.exit (0);
                }
            });

            // Layout the frame with the details pane.
            f.getContentPane ().setLayout (new BorderLayout ());
            f.getContentPane ().add ("Center", detailsPane);
            f.pack ();
            f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
           System.exit (0);
        }
    }


}
```

範例：使用 VIFSDirectory

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////
//
// VIFSDirectory example.This program presents a tree view of
// some directories in the integrated file system.
//
// Command syntax:
//VIFSDirectoryExample system
//
// This source is an example of IBM Toolbox for Java "VIFSDirectory".
//
///////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
```

```
import java.awt.event.*;

public class VIFSDirectoryExample
{


    public static void main (String[] args)
    {
        // If a system was not specified, then display help text and
        // exit.
        if (args.length != 1)
        {
            System.out.println("Usage:  VIFSDirectoryExample system");
            return;
        }

        try
        {
            // Create an AS400 object.  The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);

            // Create a VIFSDirectory object which represents the root
            // of the directory tree that we are going to show.
            VIFSDirectory directory = new VIFSDirectory (system, "/QIBM/ProdData");

            // Create a frame.
            JFrame f = new JFrame ("VIFSDirectory example");

            // Create an error dialog adapter.This will display
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

            // Create a tree pane to present the directories hierarchically.
            // Load the information from the system.
            AS400TreePane treePane = new AS400TreePane (directory);
            treePane.addErrorListener (errorHandler);

treePane.load ();

            // When the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
                public void windowClosing (WindowEvent event)
                {
                    System.exit (0);
                }
            });

            // Layout the frame with the tree pane.
            f.getContentPane ().setLayout (new BorderLayout ());
            f.getContentPane ().add ("Center", treePane);
            f.pack ();
            f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
           System.exit (0);
        }
    }


}
```

範例：使用 VPrinters

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// VPrinters example.This program presents various network
// print resources with an explorer pane.
//
// Command syntax:
//VPrintersExample system
//
// This source is an example of IBM Toolbox for Java "VPrinters".
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VPrintersExample
{


 public static void main(String[] args)
    {
        // If a system was not specified, then display help text and
        // exit.
        if (args.length != 1)
        {
            System.out.println("Usage:  VPrintersExample system");
            return;
        }

        try
        {
            // Create an AS400 object.  The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);

            // Create a VPrinters object which represents the list
// of printers attached to the system.
VPrinters printers = new VPrinters (system);


            // Create a frame.
            JFrame f = new JFrame ("VPrinters example");

            // Create an error dialog adapter.This will display
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

            // Create an explorer pane to present the network print resources.
            // Use load to load the information from the system.
            AS400ExplorerPane explorerPane = new AS400ExplorerPane (printers);
            explorerPane.addErrorListener (errorHandler);

explorerPane.load ();

            // When the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
                public void windowClosing (WindowEvent event)
                {
                    System.exit (0);
                }
            });
```

```
            // Layout the frame with the explorer pane.
            f.getContentPane ().setLayout (new BorderLayout ());
            f.getContentPane ().add ("Center", explorerPane);
            f.pack ();
            f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
           System.exit (0);
        }
    }


}
```

範例：使用 CommandCallMenuItem

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////
//
// Command call menu item example.This program demonstrates how to
// use a menu item that calls a server command.It will display
// any messages that are returned in a dialog.
//
// Command syntax:
//CommandCallMenuItemExample system
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class CommandCallMenuItemExample
{



    private static JFrame f;



     public static void main(String[] args)
    {
        // If a system was not specified, then display help text and
        // exit.
        if (args.length != 1)
        {
            System.out.println("Usage:  CommandCallMenuItemExample system");
                return;
        }

         try
        {
             // Create an AS400 object.
The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);

            // Create a frame.
            f = new JFrame ("Command call menu item example"

            // Create an error dialog adapter.This will display
```

```
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

            // Create a CommandCallMenuItem object to run the command.
            CommandCallMenuItem menuItem =
               new CommandCallMenuItem ("Clear library FRED", null, system, "CLRLIB FRED");
            menuItem.addErrorListener (errorHandler);

            // Add an action completed listener to display any
            // returned messages in a dialog.
             menuItem.addActionCompletedListener (new ActionCompletedListener ()
            {
                public void actionCompleted (ActionCompletedEvent event)
                {
                    // Get the message list from the event source.
                    CommandCallMenuItem item = (CommandCallMenuItem) event.getSource ();
                    AS400Message[] messageList = item.getMessageList ();

                    // Use an AS400DetailsPane to display the messages.
                    VMessageList vmessageList = new VMessageList (messageList);
                    AS400DetailsPane messageDetails = new AS400DetailsPane (vmessageList);
                    messageDetails.load ();

                    // Show the details in a dialog.
                    JDialog dialog = new JDialog(f);
                    dialog.getContentPane().setLayout(new BorderLayout());
                    dialog.getContentPane().add("Center"messageDetails);
                    dialog.pack();
                    dialog.setVisible(true);
                }
            });

            // Create a menu with the item.
            JMenu menu = new JMenu ("Server Command Calls");
             menu.add (menuItem);

            JMenuBar menuBar = new JMenuBar ();
            menuBar.add (menu);

            f.getRootPane ().setJMenuBar (menuBar);

            // When the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
                public void windowClosing (WindowEvent event)
                {
                     System.exit(0);
                }
            });

            // Layout the frame with the details pane.
            f.getContentPane ().setLayout (new BorderLayout ());
            f.setSize (300, 400);
            f.show ();
        }
        catch (Exception e)
        {
                            System.out.println ("Error: " + e.getMessage ());
            System.exit(0);
        }
    }


}
```

範例：使用 DataQueueDocument

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
////////////////////////////////////////////////////////////////////////
//
// Data queue document example.This program demonstrates how to
// use a document that is associated with a server data queue.
//
// Command syntax:
//DataQueueDocumentExample system read|write
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class DataQueueDocumentExample
{
    private static DataQueueDocumentdqDocument;
    private static JTextField text;
    private static booleanrw;

     public static void main(String[] args)
    {
        // If a system or read|write was not specified, then display
        // help text and exit.
        if (args.length != 2)
        {
            System.out.println("Usage:  DataQueueDocumentExample system read|write");
            return;
        }

        rw = args[1].equalsIgnoreCase ("read");
        String mode = rw ? "Read" : "Write";

        try
        {
            // Create two frames.
            JFrame f =
               new JFrame ("Data queue document example - " + mode);

            // Create an error dialog adapter.This will display
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

            // Create a working cursor adapter.This will adjust
// the cursor whenever a data queue is read or written.
            WorkingCursorAdapter cursorAdapter = new WorkingCursorAdapter (f);

            // Create an AS400 object.  The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);

            // Create the data queue path name.
            QSYSObjectPathName dqName = new QSYSObjectPathName ("QGPL", "JAVATALK", "DTAQ");

            // Make sure the the data queue exists.
            DataQueue dq = new DataQueue (system, dqName.getPath ());
            try
            {
                dq.create (200);
            }
            catch (Exception e)
```

```
            {
                // Ignore exceptions.Most likely, the data queue
// already exists.
            }

            // Create a DataQueueDocument object.
            dqDocument = new DataQueueDocument (system, dqName.getPath ());
            dqDocument.addErrorListener (errorHandler);
            dqDocument.addWorkingListener (cursorAdapter);

            // Create a text field used to present the document.
            text = new JTextField (dqDocument, "", 40);
            text.setEditable (! rw);
// When the program runs, we need a way to control when
            // the reads and writes take place.We will let the
// use control this with a button.
            Button button = new Button (mode);
            button.addActionListener (new ActionListener ()
            {
                public void actionPerformed (ActionEvent event)
                {
                    if (rw)
                        dqDocument.read ();
                    else {
                        dqDocument.write ();
                        text.setText ("");
                    }
                }
            });

            // When the the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
                public void windowClosing (WindowEvent event)
                {
                    System.exit (0);
                }
            });

            // Layout the frame.
            f.getContentPane ().setLayout (new FlowLayout ());
            f.getContentPane ().add (text);
            f.getContentPane ().add (button);
            f.pack ();
            f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
            System.exit(0);
        }
    }
}
```

範例：使用 IFSFileDialog

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
///////////////////////////////////////////////////////////////////////////
//
// File Dialog example.
//
///////////////////////////////////////////////////////////////////////////

import java.io.*;
import java.awt.*;
import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
```

```
public class FileDialogExample extends Object
{

   public static void main(String[] parameters)
   {

            System.out.println( " " );



      // if a system name was not specified, display help text and exit.

      if (parameters.length >= 1)
      {

          // The first parameter is the system that contains the files.

         String system = parameters[0];



          try
         {
             // Create an AS400 object for the server that contains the files.
             // Connect to the file server on the server.  Connect now so
             // the sign-on screen is displayed now.

             AS400 as400 = new AS400(system);
             as400.connectService(AS400.FILE);



             // Create a frame to hold the dialog.

             Frame frame = new Frame();



             // Create the file dialog object.

             IFSFileDialog fileDialog = new IFSFileDialog(frame, "File Open", as400);



             // Create the list of filters the user can choose then add the filters
             // to the dialog.

             FileFilter[] filterList = {
                new FileFilter("All files (*.*)", "*.*"),
                new FileFilter("Executables (*.exe)", "*.exe"),
                new FileFilter("HTML files (*.html)", "*.html"),
                new FileFilter("Images (*.gif)", "*.gif"),
                new FileFilter("Text files (*.txt)", "*.txt")};

             fileDialog.setFileFilter(filterList, 0);



             // Set the text for the "OK" button on the dialog.

             fileDialog.setOkButtonText("Open");



             // Set the text for the "Cancel" button on the dialog.
```

```
        fileDialog.setCancelButtonText("Cancel");


        // Set the initial directory for the dialog.

        fileDialog.setDirectory("/");


        // Display the dialog and wait until the user presses OK or Cancel

        int pressed = fileDialog.showDialog();


        // If the user pressed OK, get the fully qualified path and name
        // of the file they chose.

        if (pressed == IFSFileDialog.OK)
        {
           System.out.println("User selected: " +
                              fileDialog.getAbsolutePath());
        }


        // Else if the user pressed cancel, display a message.

        else if (pressed == IFSFileDialog.CANCEL)
        {
           System.out.println("User pressed cancel");
        }

        else
           System.out.println("User didn't press Open or Cancel");

    }
    catch (Exception e)
    {
       // If any of the above operations failed say the dialog operation
       // failed and output the exception.

       System.out.println("Dialog operation failed");
       System.out.println(e);
    }
}


// Display help text when parameters are incorrect.

else
{
   System.out.println("");
   System.out.println("");
   System.out.println("");
   System.out.println("Parameters are not correct.  Command syntax is:");
   System.out.println("");
   System.out.println("  FileDialogExample system");
   System.out.println("");
   System.out.println("Where");
   System.out.println("");
   System.out.println("  system = iSeries server");
   System.out.println("");
   System.out.println("For example:");
   System.out.println("");
   System.out.println("");
```

```
         System.out.println("  FileDialogExample mySystem");
         System.out.println("");
         System.out.println("");
      }

      System.exit(0);
   }
}
```

範例：使用 IFSTextFileDocument

註: 相關的重要法律資訊，請閱讀程式碼範例免責聲明。

```
///////////////////////////////////////////////////////////////////////////
//
// IFS text file document example.This program demonstrates how to
// use a document that is associated with a text file in the AS/400
// integrated file system.
//
// Command syntax:
//IFSTextFileDocumentExample system path
//
///////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class IFSTextFileDocumentExample
{



    private static IFSTextFileDocumentdocument;
    private static JTextPanetext;



     public static void main(String[] args)
    {
        // If a system or path was not specified, then display
        // help text and exit.
        if (args.length != 2)
        {
            System.out.println("Usage:  IFSTextFileDocumentExample system path");
            return;
        }

        try
        {
            // Create two frames.
            JFrame f = new JFrame ("IFS text file document example");

            // Create an error dialog adapter.This will display
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

            // Create a working cursor adapter.This will adjust
// the cursor whenever the text file is read or written.
            WorkingCursorAdapter cursorAdapter = new WorkingCursorAdapter (f);

            // Create an AS400 object.  The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);
```

```
            // Create and load the IFS text file document.
            document = new IFSTextFileDocument (system, args[1]);
            document.addErrorListener (errorHandler);
            document.addWorkingListener (cursorAdapter);
            document.load ();

            // Create the text pane used to present the document.
            text = new JTextPane (document);
            text.setSize (new Dimension (500, 500));

            // Set up a scroll pane to use with the text pane.
            JScrollPane scroll = new JScrollPane (text);
            scroll.setHorizontalScrollBarPolicy (JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
            scroll.setVerticalScrollBarPolicy (JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

            // Create a menu bar with a single menu.
            MenuBar menuBar = new MenuBar ();
            Menu menu = new Menu ("File");
            menuBar.add (menu);

            // Add menu items to load and save.
            MenuItem load = new MenuItem ("Load");
            load.addActionListener (new ActionListener ()
            {
                public void actionPerformed (ActionEvent event)
                {
                    document.load ();
                }
            });
            menu.add (load);

            MenuItem save = new MenuItem ("Save");
            save.addActionListener (new ActionListener ()
            {
                public void actionPerformed (ActionEvent event)
                {
                    document.save ();
                }
            });
            menu.add (save);

            // When the the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
                public void windowClosing (WindowEvent event)
                {
                    System.exit (0);
                }
            });

            // Layout the frame.
            f.getContentPane ().setLayout (new BorderLayout ());
            f.getContentPane ().add ("Center", scroll);
            f.setMenuBar (menuBar);
            f.pack ();
            f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
           System.exit (0);
        }
    }

}
```

AS400 JDBCDataSourcePane

AS400JDBCDataSourcePane 類別表示 AS400JDBCDataSource 物件的內容值。並可選擇性地對 AS400JDBCDataSource 物件進行變更。

AS400JDBCDataSourcePane 延伸 JComponent。若要使用 AS400JDBCDataSourcePane 來顯示資料來源的內容，可在 AS400JDBCDataSourcePane 建構子上指定資料來源，或是於 使用 setDataSource() 建立 AS400JDBCDataSourcePane 之後指定資料來源。 使用 applyChanges()，可將對圖形式使用者介面 (GUI) 所做的變更套用到資料來源物件。

範例：使用 AS400JDBCDataSourcePane

下列範例建立 AS400JDBCDataSourcePane 及**確定**按鈕，並將之新增到框架。當您按一下**確定**時，便可將對 GUI 的變更套用到資料來源。

```
                // Create a data source.
                myDataSource = new AS400JDBCDataSource();

                // Create a window to hold the pane and an OK button.
                JFrame frame = new JFrame ("JDBC Data Source Properties");

                // Create a data source pane.
                dataSourcePane = new AS400JDBCDataSourcePane(myDataSource);

                // Create an OK button
                JButton okButton = new JButton("OK");

                // Add an ActionListener to the OK button.  When OK is
        // pressed, applyChanges() will be called to commit any
                // changes to the data source.
                okButton.addActionListener(new ActionListener()
           {
              public void actionPerformed(ActionEvent ev)
              {
                 // Apply all changes made on the data source pane
                 // to the data source.  If all changes are applied
                 // successfully, get the data source from the pane.
                 if (dataSourcePane.applyChanges())
                 {
                    System.out.println("ok pressed");
                    myDataSource = dataSourcePane.getDataSource();
                    System.out.println(myDataSource.getServerName());
                 }
              }
           }
        );

        // Setup the frame to show the pane and OK button.
        frame.getContentPane ().setLayout (new BorderLayout ());
        frame.getContentPane ().add ("Center", dataSourcePane);
        frame.getContentPane ().add ("South", okButton);

        // Pack the frame.
        frame.pack ();

        //Display the pane and OK button.
        frame.show ();
```

範例：使用 VJobList 顯示工作清單

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////
//
// Job list example.This program presents a list of jobs in an
// explorer pane.
//
// Command syntax:
//VJobListExample system
//
// This source is an example of IBM Toolbox for Java "AS400ExplorerPane".
//
///////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VJobListExample
{


    public static void main (String[] args)
    {
        // If a system was not specified, then display help text and
        // exit.
        if (args.length != 1)
        {
            System.out.println("Usage:  VJobListExample system");
            return;
        }

        try
        {
            // Create an AS400 object.  The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);

            // Create a VJobList object which represents the list
            // of jobs named QZDASOINIT.
            VJobList jobList = new VJobList (system);
            jobList.setName ("QZDASOINIT");

            // Create a frame.
            JFrame f = new JFrame ("Job list example");

            // Create an error dialog adapter.This will display
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

            // Create an explorer pane to present the job list.
            // Use load to load the information from the system.
            AS400ExplorerPane explorerPane = new AS400ExplorerPane (jobList);
            explorerPane.addErrorListener (errorHandler);

explorerPane.load ();

            // When the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
                public void windowClosing (WindowEvent event)
                {
                    System.exit (0);
                }
            });

            // Layout the frame with the explorer pane.
            f.getContentPane ().setLayout (new BorderLayout ());
```

```
            f.getContentPane ().add ("Center", explorerPane);
            f.pack ();
            f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
           System.exit (0);
        }
    }


}
```

範例：使用 VMessageQueue

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// Message queue example.This program presents a message queue in an
// explorer pane.
//
// Command syntax:
//VMessageQueueExample system
//
// This source is an example of IBM Toolbox for Java "VMessageQueue".
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VMessageQueueExample
{


    public static void main (String[] args)
    {
        // If a system was not specified, then display help text and
        // exit.
        if (args.length != 1)
        {
            System.out.println("Usage:  VMessageQueueExample system");
            return;
        }

        try
        {
            // Create an AS400 object.  The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);

            // Force the user to sign on so that we know the user id.
            system.connectService (AS400.COMMAND);

            // Create a VMessageQueue object which represents the
            // current user's message queue.
            VMessageQueue queue = new VMessageQueue (system,
                QSYSObjectPathName.toPath ("QUSRSYS", system.getUserId (),
                "MSGQ"));

            // Create a frame.
            JFrame f = new JFrame ("Message queue example");
```

```
            // Create an error dialog adapter.This will display
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

            // Create an explorer pane to present the message queue.
            // Use load to load the information from the system.
            AS400ExplorerPane explorerPane = new AS400ExplorerPane (queue);
            explorerPane.addErrorListener (errorHandler);

explorerPane.load ();

            // When the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
                public void windowClosing (WindowEvent event)
                {
                    System.exit (0);
                }
            });

            // Layout the frame with the explorer pane.
            f.getContentPane ().setLayout (new BorderLayout ());
            f.getContentPane ().add ("Center", explorerPane);
            f.pack ();
            f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
           System.exit (0);
        }
    }


}
```

範例：使用按鈕呼叫伺服器上的程式

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////
//
// Program call button example.This program demonstrates how to
// use a button that calls a program on the server.It will exchange data
// with the server program via an input and output parameter.
//
// Command syntax:
//ProgramCallButtonExample system
//
// This source is an example of IBM Toolbox for Java "ProgramCallButton".
//
///////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class ProgramCallButtonExample
{

    private ProgramParameter parm1, parm2, parm3, parm4, parm5;
    private JTextField cpuField;
    private JTextField dasdField;
    private JTextField jobsField;
```

```
    // Create a ProgramCallButtonExample object, then call the
    // non-static version of main().If we don't to this then
// the class variables (parm1, parm2, ...) must be declared
// static.If they are static they cannot be used by the
// action completed listener in Java 1.1.7 or 1.1.8.
     public static void main(String[] args)
    {
        ProgramCallButtonExample me = new ProgramCallButtonExample();
        me.Main(args);
    }

    public void Main (String[] args)
    {
        // If a system was not specified, then display help text and
        // exit.
        if (args.length != 1)
        {
            System.out.println("Usage:  ProgramCallButtonExample system");
            return;
        }

        try
        {
            // Create a frame.
            JFrame f = new JFrame ("Program call button example");

            // Create an error dialog adapter.This will display
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

            // Create an AS400 object.  The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);

            // Create the program path name.
            QSYSObjectPathName programName = new QSYSObjectPathName ("QSYS",
                "QWCRSSTS", "PGM");

            // Create a ProgramCallButton object.  The button
            // will have the text "Refresh" and no icon.
            ProgramCallButton button = new ProgramCallButton ("Refresh", null);
            button.setSystem (system);
            button.setProgram (programName.getPath ());
            button.addErrorListener (errorHandler);

            // The first parameter is an 64 byte output parameter.
            parm1 = new ProgramParameter (64);
            button.addParameter (parm1);

            // We use the second parameter to set the buffer size
            // of the first parameter.We will always set this to
// 64.Remember that we need to convert the Java int
// value 64 to the format used on the server.
            AS400Bin4 parm2Converter = new AS400Bin4 ();
            byte[] parm2Bytes = parm2Converter.toBytes (64);
            parm2 = new ProgramParameter (parm2Bytes);
            button.addParameter (parm2);

            // The third parameter is the status format.We will
// always use "SSTS0200".  This is a String value, and
            // again we need to convert it to the format used on the server.
            AS400Text parm3Converter = new AS400Text (8, system);
            byte[] parm3Bytes = parm3Converter.toBytes ("SSTS0200");
            parm3 = new ProgramParameter (parm3Bytes);
            button.addParameter (parm3);
```

```
            // The fourth parameter is the reset statistics parameter.
            // We will always pass "*NO" as a 10 character String.
            AS400Text parm4Converter = new AS400Text (10, system);
            byte[] parm4Bytes = parm4Converter.toBytes ("*NO       ");
            parm4 = new ProgramParameter (parm4Bytes);
            button.addParameter (parm4);

            // The fifth parameter is for error information.It
// is an input/output parameter.We will not use it
// for this example, but we need to set it to something,
            // or else the number of parameters will not match
            // what the server is expecting.
            byte[] parm5Bytes = new byte[32];
            parm5 = new ProgramParameter (parm5Bytes, 0);
            button.addParameter (parm5);

            // When the program runs, we will get a bunch of data.
            // We need a way to display that data to the user.
            // In this case, we will just use simple labels and text
            // fields.
            JLabel cpuLabel = new JLabel ("CPU Utilitization: ");
            cpuField = new JTextField (10);
            cpuField.setEditable (false);

            JLabel dasdLabel = new JLabel ("DASD Utilitization: ");
            dasdField = new JTextField (10);
            dasdField.setEditable (false);

            JLabel jobsLabel = new JLabel ("Number of active jobs: ");
            jobsField = new JTextField (10);
            jobsField.setEditable (false);

            // When the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
public void windowClosing (WindowEvent event)
                {
                    System.exit (0);
                }
            });

            // When the program is called, we need to process the
            // information that comes back in the first parameter.
            // The format of the data in this parameter was documented
            // by the program we are calling.

button.addActionCompletedListener (new ActionCompletedListener ()
            {
                public void actionCompleted (ActionCompletedEvent event)
                {
                    try
                    {
                       // Get the data from the first parameter.
                       // It is in the server format.
                       byte[] parm1Bytes = parm1.getOutputData ();

                       // Each of the pieces of data that we need
                       // is an int.  We can create one converter
                       // to do all of our conversions.
                       AS400Bin4 parm1Converter = new AS400Bin4 ();

                       // Get the CPU utilitization starting at byte 32.
                       // Set this value in the corresponding text field.
                       int cpu = parm1Converter.toInt (parm1Bytes, 32);
                       cpuField.setText (Integer.toString (cpu / 10) + "%");

                       // Get the DASD utilitization starting at byte 52.
                       // Set this value in the corresponding text field.
```

```
                    int dasd = parm1Converter.toInt (parm1Bytes, 52);
                    dasdField.setText (Integer.toString (dasd / 10000) + "%");

                    // Get the number of active jobs starting at byte 36.
                    // Set this value in the corresponding text field.
                    int jobs = parm1Converter.toInt (parm1Bytes, 36);
                    jobsField.setText (Integer.toString (jobs));
                 }
                 catch (Exception e) { e.printStackTrace(); }
              }
           });

           // Layout the frame.
           JPanel outputPanel = new JPanel ();
           outputPanel.setLayout (new GridLayout (3, 2, 5, 5));
           outputPanel.add (cpuLabel);
           outputPanel.add (cpuField);
           outputPanel.add (dasdLabel);
           outputPanel.add (dasdField);
           outputPanel.add (jobsLabel);
           outputPanel.add (jobsField);

           Panel buttonPanel = new Panel ();
           buttonPanel.add (button);

           f.getContentPane ().setLayout (new BorderLayout ());
           f.getContentPane ().add ("Center", outputPanel);
           f.getContentPane ().add ("South", buttonPanel);
           f.pack ();
           f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
           System.exit (0);
        }
    }


}
```

範例：使用 VPrinter

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////////
//
// VPrinter example.This program presents a printer and its spooled
// files in an explorer pane.
//
// Command syntax:
//VPrinterExample system
//
///////////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VPrinterExample
{


    public static void main (String[] args)
    {
```

```
        // If the user does not supply a printer name then show printer information
        // for a printer called OS2VPRT;
        String printerName = "OS2VPRT";

        // If a system was not specified, then display help text and
        // exit.
        if (args.length == 0)
        {
            System.out.println("Usage:VPrinterExample system printer");
            return;
        }


        // If the user specified a name, use it instead of the default.
        if (args.length > 1)
           printerName = args[1];


        try
        {
            // Create an AS400 object.  The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);


            // Create a Printer object (from the Toolbox access package)
            // which represents the printer, then create a VPrinter
            // object to graphically show the spooled files on the printer.
            Printer printer = new Printer(system, printerName);
            VPrinter vprinter = new VPrinter(printer);

            // Create a frame to hold our window.
            JFrame f = new JFrame ("VPrinter Example");

            // Create an error dialog adapter.This will display
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);


            // Create an explorer pane to present the printer and its spooled
            // files.Use load to load the information from the system.
AS400ExplorerPane explorerPane = new AS400ExplorerPane (vprinter);
            explorerPane.addErrorListener (errorHandler);

explorerPane.load ();


            // When the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
                public void windowClosing (WindowEvent event)
                {
                    System.exit (0);
                }
            });

            // Layout the frame with the explorer pane.
            f.getContentPane ().setLayout (new BorderLayout ());
            f.getContentPane ().add ("Center", explorerPane);
            f.pack ();
            f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
```

```
           System.exit (0);
        }
    }
}
```

範例：使用 VPrinters

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// VPrinters example.This program presents various network
// print resources with an explorer pane.
//
// Command syntax:
//VPrintersExample system
//
// This source is an example of IBM Toolbox for Java "VPrinters".
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VPrintersExample
{


public static void main (String[] args)
    {
        // If a system was not specified, then display help text and
        // exit.
        if (args.length != 1)
        {
            System.out.println("Usage:  VPrintersExample system");
            return;
        }

        try
        {
            // Create an AS400 object.  The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);

            // Create a VPrinters object which represents the list
// of printers attached to the system.
VPrinters printers = new VPrinters (system);


            // Create a frame.
            JFrame f = new JFrame ("VPrinters example");

            // Create an error dialog adapter.This will display
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

            // Create an explorer pane to present the network print resources.
            // Use load to load the information from the system.
            AS400ExplorerPane explorerPane = new AS400ExplorerPane (printers);
            explorerPane.addErrorListener (errorHandler);

explorerPane.load ();

            // When the frame closes, exit.
```

```
            f.addWindowListener (new WindowAdapter () {
                public void windowClosing (WindowEvent event)
                {
                    System.exit (0);
                }
            });

            // Layout the frame with the explorer pane.
            f.getContentPane ().setLayout (new BorderLayout ());
            f.getContentPane ().add ("Center", explorerPane);
            f.pack ();
            f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
           System.exit (0);
        }
    }


}
```

VPrinterOutput 範例

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////////
//
// VPrinterOutput example.This program presents a list of spooled
// files on the server.All spooled files, or spooled files for
// a specific user can be displayed.
//
// Command syntax:
//    VPrinterOutputExample system <user>
//
//(User is optional, if not specified all spooled files on the system
//will be displayed.Caution - listing all spooled files on the system
//and take a long time)
//
////////////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VPrinterOutputExample
{


    public static void main (String[] args)
    {

        // If a system was not specified, display help text and exit.
        if (args.length == 0)
        {
            System.out.println("Usage:  VPrinterOutputExample system <user>");
            return;
        }

        try
        {
            // Create an AS400 object.  The system name was passed
            // as the first command line argument.
            AS400 system = new AS400 (args[0]);
```

```
            system.connectService(AS400.PRINT);

            // Create the VPrinterOutput object.
            VPrinterOutput printerOutput = new VPrinterOutput(system);

            // If a user was specified as a command line parameter, tell
            // the printerObject to get spooled files only for that user.
            if (args.length > 1)
               printerOutput.setUserFilter(args[1]);


            // Create a frame to hold our window.
            JFrame f = new JFrame ("VPrinterOutput Example");

            // Create an error dialog adapter.This will display
// any errors to the user.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);


            // Create an details pane to present the list of spooled files.
            // Use load to load the information from the system.
            AS400DetailsPane detailsPane = new AS400DetailsPane (printerOutput);
            detailsPane.addErrorListener (errorHandler);

detailsPane.load ();


            // When the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
                public void windowClosing (WindowEvent event)
                {
                    System.exit (0);
                }
            });

            // Layout the frame with the details pane.
            f.getContentPane ().setLayout (new BorderLayout ());
            f.getContentPane ().add ("Center", detailsPane);
            f.pack ();
            f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
           System.exit (0);
        }
    }
}
```

範例：使用 SQLQueryBuilderPane

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// SQLQueryBuilderPane example.This program presents a query builder
// which allows the user to build a SQL query.
//
// Command syntax:
//SQLQueryBuilderPaneExample system
//
// This source is an example of IBM Toolbox for Java "SQLQueryBuilderPane",
// and "SQLResultSetFormPane".
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
```

```
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class SQLQueryBuilderPaneExample
{


    // This connection is shared by all components.
    private SQLConnection connection;



    // This error handler is shared by all components.
    private ErrorDialogAdapter errorHandler;



    // The query builder pane.
    private SQLQueryBuilderPane queryBuilderPane;



    // This is the main java calls.Here we create an instance of our
// class and call our own Main() method.We do this to avoid
// problems with static.Java has some restrictions with static
// methods using non-static data, especially when it comes to
    // inner classes.The code is cleaner if we keep the amount of
// static data and methods to a minimum.
     public static void main(String[] args)
    {
        SQLQueryBuilderPaneExample me = new SQLQueryBuilderPaneExample();
        me.Main(args);
    }


    public void Main (String[] args)
    {
        // If a system was not specified, then display
        // help text and exit.
        if (args.length != 1)
        {
            System.out.println("Usage:  SQLQueryBuilderPaneExample system");
            return;
        }

        try
        {
            // Register the IBM Toolbox for Java JDBC driver.
            DriverManager.registerDriver(new AS400JDBCDriver());

            // Create an SQLConnection object.  The system name was passed
            // as the first command line argument.
            connection = new SQLConnection ("jdbc:as400://" + args[0]);

            // Create a frame.
            JFrame f = new JFrame ("SQLQueryBuilderPane example");

            // Create an error dialog adapter.This will display
// any errors to the user.
            errorHandler = new ErrorDialogAdapter (f);

            // Create a SQL query builder pane to present the query
            // builder.  Load the data that is needed for the query
            // builder from the system.
```

```
            queryBuilderPane = new SQLQueryBuilderPane (connection);
            queryBuilderPane.addErrorListener (errorHandler);
            queryBuilderPane.load ();

            // Create a button which will display the results of
            // the generated query in a form pane in another frame.
            JButton resultSetButton = new JButton ("Show result set");
            resultSetButton.addActionListener (new ActionListener ()
            {
                public void actionPerformed (ActionEvent event)
                {
                    showFormPane (queryBuilderPane.getQuery ());
                }
            });

            // When the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
public void windowClosing (WindowEvent event)
                {
                    System.exit (0);
                }
            });

            // Layout the frame with the query builder pane.
            f.getContentPane ().setLayout (new BorderLayout ());
            f.getContentPane ().add ("Center", queryBuilderPane);
            f.getContentPane ().add ("South", resultSetButton);
            f.pack ();
            f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
           System.exit (0);
        }
    }



    private void showFormPane (String query)
    {
        // Create a new frame for the results of the query.
        JFrame f = new JFrame (query);

        // Create a SQL result set form pane to present the results
        // of the query.Load the results from the system.
SQLResultSetFormPane formPane = new SQLResultSetFormPane (connection, query);
        formPane.addErrorListener (errorHandler);
        formPane.load ();

        // Layout the frame with the form pane.
        f.getContentPane ().setLayout (new BorderLayout ());
        f.getContentPane ().add ("Center", formPane);
        f.pack ();
        f.show ();
    }


}
```

範例：使用 SQLResultSetTablePane

註: 請閱讀程式碼範例免責聲明中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// SQLResultSetTablePane example.This program presents the contents of
```

```
// a table in a table pane.There is a SQLStatementDocument which allows
// the user to type in any SQL statement.In addition, there is a button
// which allows the user to delete all rows of the table.
//
// Command syntax:
//SQLResultSetTablePaneExamplesystemtable
//
// This source is an example of IBM Toolbox for Java "SQLQueryBuilderPane",
// "SQLResultSetFormPane", and "SQLStatementButton".
//
//////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class SQLResultSetTablePaneExample
{



    private static SQLStatementDocument document;
    private static SQLResultSetTablePanetablePane;



     public static void main(String[] args)
    {
        // If a system was not specified, then display
        // help text and exit.
        if (args.length != 2)
        {
            System.out.println("Usage:  SQLResultSetTablePaneExample  system  table");
            return;
        }

        try
        {
            // Register the IBM Toolbox for Java JDBC driver.
            DriverManager.registerDriver(new AS400JDBCDriver());

            // Create an SQLConnection object.  The system name was passed
            // as the first command line argument.
This connection is
            // shared by all components.
            SQLConnection connection = new SQLConnection ("jdbc:as400://" + args[0]);

            // Create a frame.
            JFrame f = new JFrame ("SQLResultSetTablePane example");

            // Create an error dialog adapter.This will display
// any errors to the user.
This error handler is shared
            // by all components.
            ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

            // Create a SQL statement document which allows the
            // user to enter a query.
            document = new SQLStatementDocument (connection, "");
            document.addErrorListener (errorHandler);

            // Create a text field for presenting the document.
            JTextField textField = new JTextField (document,
                "Enter a SQL statement here.", 50);
```

```
            // Create a button that deletes all rows of the table.
            SQLStatementButton deleteAllButton = new SQLStatementButton ("Delete all rows");
            deleteAllButton.setConnection (connection);
            deleteAllButton.setSQLStatement ("DELETE FROM " + args[1]);
            deleteAllButton.addErrorListener (errorHandler);

            // Create a SQL result set table pane to present the results
            // of a query.Load the contents immediately.
tablePane = new SQLResultSetTablePane (connection, "SELECT * FROM " + args[1]);
            tablePane.addErrorListener (errorHandler);
            tablePane.load ();

            // When enter is pressed in the text field,
            // execute the SQL statement and update the table pane.
            textField.addKeyListener (new KeyAdapter ()
            {
                public void keyPressed (KeyEvent event)
                {
                    if (event.getKeyCode () == KeyEvent.VK_ENTER)
                    {
                        // If the SQL statement is a SELECT, then
                        // let the table pane execute it, otherwise,
                        // let the document execute it.
                        String sql = document.getSQLStatement ();
                        if (sql.toUpperCase ().startsWith ("SELECT"))
                        {
                            try
                            {
                                tablePane.setQuery (sql);
                            }
                            catch (Exception e)
                            {
                                // Ignore.
                            }
                            tablePane.load ();
                        }
                        else
                            document.execute ();
                    }
                }
            });

            // When all rows are deleted using the button, then
            // update the table pane.
            deleteAllButton.addActionCompletedListener (new ActionCompletedListener ()
            {
                public void actionCompleted (ActionCompletedEvent event)
                {
                    tablePane.load ();
                }
            });

            // When the frame closes, exit.
            f.addWindowListener (new WindowAdapter () {
public void windowClosing (WindowEvent event)
                {
                    System.exit (0);
                }
            });

            // Layout the frame with the query builder pane.
            f.getContentPane ().setLayout (new BorderLayout ());
            f.getContentPane ().add ("North", textField);
            f.getContentPane ().add ("Center", tablePane);
            f.getContentPane ().add ("South", deleteAllButton);
            f.pack ();
```

```
            f.show ();
        }
        catch (Exception e)
        {
           System.out.println ("Error: " + e.getMessage ());
           System.exit (0);
        }
    }


}
```

範例：XPCML

本節將列出 IBM Toolbox for Java XPCML 元件之文件所提供的所有程式碼範例。

- 第 694 頁的『範例：壓縮現有的 XPCML 文件』
- 第 694 頁的『範例：壓縮現有的 XPCML 文件』
- 第 697 頁的『範例：使用壓縮的 XPCML 來建立 ProgramCallDocument 物件』
- 第 697 頁的『範例：取得程式呼叫結果作爲壓縮的 XPCML』
- 『範例：擷取程式呼叫結果作爲 XPCML』
- 第 691 頁的『範例：傳入參數值作爲 XPCML』
- 第 692 頁的『範例：傳入參數值陣列作爲 XPCML』
- 第 393 頁的『範例：轉換 PCML 文件至 XPCML 文件』

下列免責聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼範例免責聲明

IBM 授予您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的功能，來符合您的特定需要。

IBM 提供的所有範例程式碼僅供說明之用。這些範例尚未徹底經過所有情況的測試。因此，IBM 不擔保或默示這些程式的可靠性、服務性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：擷取程式呼叫結果作為 XPCML

下列範例說明如何建構 XPCML ProgramCallDocument、呼叫 iSeries 程式，以及擷取程式呼叫結果作爲 XPCML。此範例以下列元件爲前提：

- XPCML 文件 qgyolaus.xpcml，它能以輸入值來定義程式與參數規格
- Java 程式碼，可建構 ProgramCallDocument 物件、使用 XPCML 檔案，然後呼叫程式 QGYOLAUS
- 程式呼叫結果，由 Java 程式碼產生作爲 XPCML，儲存於檔案 XPCMLOut.xpcml 中

注意原始檔以及產生之 XPCML 中陣列資料的指定方式。元素 qgyolaus.receiver，一個輸出參數， 是 XPCML arrayOfStructParm，它具有能爲 listInfo.rcdsReturned 設定計數的屬性。下列範例程式碼只包含了部分的 QGYOLAUS 輸出。如果範例包含了所有的輸出，程式碼可能會在 <arrayOfStructParm> XPCML 標籤下列出 89 個使用者。

若爲結構的陣列，XPCML 會使用 <struct_i> XPCML 標籤，以隔開每一個 structParm 元素。每一個 <struct_i> 標籤均表示，它裡面所含的資料爲類型 autu0150 結構的一個元素。<struct_i> 標籤的索引屬性可指定結構的陣列元素。

若爲簡式類型的陣列，例如 arrayOfStringParm、arrayOfIntParm 等，<i> XPCML 標籤會列出陣列元素。

XPCML 文件 qgyolaus.xpcml

```
<xpcml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="xpcml.xsd" version="4.0">

<!-- XPCML source for calling "Open List of Authorized Users" -->
<!-- (QGYOLAUS) API -->

  <!-- Format AUTU0150 - Other formats are available -->
  <struct name="autu0150">
    <stringParm name="name" length="10"/>
    <stringParm name="userOrGroup" length="1"/>
    <stringParm name="groupMembers" length="1"/>
    <stringParm name="description" length="50"/>
  </struct>

  <!-- List information structure (common for "Open List" type APIs) -->
  <struct name="listInfo">
    <intParm name="totalRcds"/>
    <intParm name="rcdsReturned">0</rcdsReturned>
    <hexBinaryParm name="rqsHandle" totalBytes="4"/>
    <intParm name="rcdLength"/>
    <stringParm name="infoComplete" length="1"/>
    <stringParm name="dateCreated" length="7"/>
    <stringParm name="timeCreated" length="6"/>
    <stringParm name="listStatus" length="1"/>
    <hexBinaryParm totalBytes="1"/>
    <unsignedIntParm name="lengthOfInfo"/>
    <intParm name="firstRecord"/>
    <hexBinaryParm totalBytes="40"/>
  </struct>

  <!-- Program QGYOLAUS and its parameter list for retrieving -->
  <!-- AUTU0150 format -->

  <program name="QGYOLAUS" path="/QSYS.lib/QGY.lib/QGYOLAUS.pgm"
     parseOrder="listInfo receiver">
   <parameterList>
     // Output values --- array of the autu0150 struct
     <arrayOfStructParm name="receiver" count="listInfo.rcdsReturned"
        passDirection="out" outputSize="receiverLength" struct="autu0150"/>
     // Input values
     <intParm name="receiverLength" passDirection="in">16384</intParm>
     <structParm name="listInfo" passDirection="out" struct="listInfo"/>
     // Input values
     <intParm name="rcdsToReturn" passDirection="in">264</intParm>
     <stringParm name="format" passDirection="in" length="10">
        AUTU0150</stringParm>
     <stringParm name="selection" passDirection="in" length="10">
        *USER</stringParm>
     <stringParm name="member" passDirection="in" length="10">
        *NONE</stringParm>
     <intParm name="errorCode" passDirection="in">0</intParm>
   </parameterList>
  </program>
```

能建構 ProgramCallDocument 物件並且使用 XPCML 來呼叫程式 QGYOLAUS 的 Java 程式碼

```
system = new AS400();
// Create a ProgramCallDocument into which to parse the file.
ProgramCallDocument xpcmlDoc =
   new ProgramCallDocument(system, "QGYOLAUS.xpcml");

// Call QGYOLAUS
boolean rc = xpcmlDoc.callProgram("QGYOLAUS");

// Obtain program call results as XPCML and store them
// in file XPCMLOut.xpcml
if (rc)     // Program was successful
   xpcmlDoc.generateXPCML("QGYOLAUS","XPCMLOut.xpcml");
```

程式呼叫結果，產生為 XPCML，並儲存於檔案 XPCMLOut.xpcml 中

```
<?xml version="1.0"?>
<xpcml version="4.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation='xpcml.xsd' >

  <program name="QGYOLAUS" path="/QSYS.lib/QGY.lib/QGYOLAUS.pgm"
     parseOrder="listInfo receiver">
  <parameterList>
      <arrayOfStructParm name="receiver" passDirection="out"
          count="listInfo.rcdsReturned" outputSize="receiverLength"
          struct="autu0150">
        <struct_i index="0">
          <stringParm name="name" length="10">JANEDOW</stringParm>
          <stringParm name="userOrGroup" length="1">0</stringParm>
          <stringParm name="groupMembers" length="1">0</stringParm>
          <stringParm name="description" length="50">
             Jane Doe</stringParm>
        </struct_i>
        <struct_i index="1">
          <stringParm name="name" length="10">BOBS</stringParm>
          <stringParm name="userOrGroup" length="1">0</stringParm>
          <stringParm name="groupMembers" length="1">0</stringParm>
          <stringParm name="description" length="50">
             Bob Smith</stringParm>
        </struct_i>

        <!-- More records here depending on how many users output. -->
        <!-- In this case 89 user records are listed here.    -->

      </arrayOfStructParm>     <!-- End of user array -->
      <intParm name="receiverLength" passDirection="in">
         16384</intParm>
      <structParm name="listInfo" passDirection="out"
         struct="listInfo">
          <intParm name="totalRcds">89</intParm>
          <intParm name="rcdsReturned">89</intParm>
          <hexBinaryParm name="rqsHandle" totalBytes="4">
             00000001==</hexBinaryParm>
          <intParm name="rcdLength">62</intParm>
          <stringParm name="infoComplete" length="1">C</stringParm>
          <stringParm name="dateCreated" length="7">
             1030321</stringParm>
          <stringParm name="timeCreated" length="6">
             120927</stringParm>
          <stringParm name="listStatus" length="1">2</stringParm>
          <hexBinaryParm totalBytes="1"></hexBinaryParm>
          <unsignedIntParm name="lengthOfInfo">
             5518</unsignedIntParm>
          <intParm name="firstRecord">1</intParm>
```

```
      </structParm>
      <intParm name="rcdsToReturn" passDirection="in">264</intParm>
      <stringParm name="format" passDirection="in" length="10">
         AUTU0150</stringParm>
      <stringParm name="selection" passDirection="in" length="10">
         *USER</stringParm>
      <stringParm name="member" passDirection="in" length="10">
         *NONE</stringParm>
      <intParm name="errorCode" passDirection="in">0</intParm>
  </parameterList>
  </program>
</xpcml>
```

範例：傳入參數值作為 XPCML

程式參數值可以在 XPCML 來源檔中設定。 當讀取並剖析 XPCML 時，會針對每一個傳入其值作爲 XPCML 的參數，自動呼叫 ProgramCallDocument setValue 方法。這可讓使用者不必再費神撰寫 Java 程式碼來爲複雜的結構與陣列設定值。

在下列範例中，XPCML 呼叫了兩個不同的程式，prog1 與 prog2。這兩個程式都使用了輸入參數 s1Ref。第一個範例會針對每一個程式呼叫設定不同的 s1Ref 值。第二個範例會針對每一個程式呼叫設定相同的 s1Ref 值，這也顯示了爲輸入參數設定固定資料值的實用方法。

範例：為輸入參數傳入不同的值

在下列範例中，當 XML 剖析器讀取並剖析文件時， 元素 prog1.s1Ref.s2Ref.s2p1[0] 的值爲 prog1Val_1，而元素 prog1.s1Ref.s2Ref.s2p1[1] 的值則爲 prog1Val_2。

```
    <xpcml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="xpcml.xsd" version="4.0">

<struct name="s1">
   <stringParm name="s1p1"/>
   <structParm name="s2Ref" struct="s2"/>
</struct>

<struct name="s2">
  <stringParm name="s2p1" length="10"/>
  <arrayOfStringParm name="parm1" count="2"/>
</struct>

<program name="prog1" path="/QSYS.LIB/W95LIB.LIB/PROG1.PGM">
 <parameterList>
   <structParm name="s1Ref" struct="s1" passDirection="in" >
     <stringParm name="s1p1">prog1Val</stringParm>
     <structParm name="s2Ref" struct="s2">
         <stringParm name="s2p1" length="10">prog1Val</stringParm>
         <arrayOfStringParm name="parm1" count="2">
            <i>prog1Val_1</i>
            <i>prog1Val_2</i>
         </arrayOfStringParm>
     </structParm>
   </structParm>
 </parameterList>
</program>

<program name="prog2" path="/QSYS.LIB/W95LIB.LIB/PROG1.PGM">
 <parameterList>
   <structParm name="s1Ref" struct="s1" passDirection="in" >
     <stringParm name="s1p1">prog2Val</stringParm>
     <structParm name="s2Ref" struct="s2">
         <stringParm name="s2p1" length="10">prog2Val</stringParm>
         <arrayOfStringParm name="parm1" count="2">
```

```
            <i>prog2Val_1</i>
            <i>prog2Val_2</i>
          </arrayOfStringParm>
        </structParm>
      </structParm>
    </parameterList>
  </program>
</xpcml>
```

範例：為輸入參數傳入常數值

在下列範例中，當 XML 剖析器讀取並剖析文件時， 元素 prog1.s1Ref.s2Ref.s2p1[0] 的值爲 constantVal_1，而元素 prog1.s1Ref.s2Ref.s2p1[1] 的值則爲 constantVal_2。

```
<xpcml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="xpcml.xsd" version="4.0">

 <struct name="s1" >
    <stringParm name="s1p1">constantVal</stringParm>
    <structParm name="s2Ref" struct="s2"/>
 </struct>

 <struct name="s2">
   <stringParm name="s2p1" length="10">constantVal</stringParm>
   <arrayOfStringParm name="parm1" count="2">
      <i>constantVal_1</i>
      <i>constantVal_2</i>
   </arrayOfStringParm>
 </struct>

 <program name="prog1" path="/QSYS.LIB/W95LIB.LIB/PROG1.PGM">
  <parameterList>
    <structParm name="s1Ref" struct="s1" passDirection="in" />
  </parameterList>
 </program>

 <program name="prog2" path="/QSYS.LIB/W95LIB.LIB/PROG1.PGM">
  <parameterList>
    <structParm name="s1Ref" struct="s1" passDirection="in" />
  </parameterList>
 </program>
</xpcml>
```

範例：傳入參數值陣列作為 XPCML

若要使用 XPCML 來傳入陣列資料，您就必須使用計數屬性：

- 在陣列元素上指定計數屬性
- 將計數屬性設定爲您在剖析文件時陣列中所含之元素數量

下列範例將說明如何透過使用 structParm 陣列資料以及結構陣列，以傳入參數值陣列。

```
<?xml version="1.0" encoding="UTF-8"?>
<xpcml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="xpcml.xsd" version="4.0">

 <struct name="s1" >
    <stringParm name="s1p1"/>
    <struct name="s1Array">
      <stringParm name="s1Ap1"/>
    </struct>
 </struct>

 <struct name="s2">
    <stringParm name="s2p1"/>
```

```
 </struct>

 <program name="prog1" path="/QSYS.LIB/W95LIB.LIB/PROG1.PGM">
  <parameterList>
    <structParm name="s1Ref" struct="s1" passDirection="in" >
        <stringParm name="s1p1">Value 1</stringParm>
        <arrayOfStruct name="s1Array" count="2">
           <struct_i>
               <stringParm name="s1Ap1">Value 1</stringParm>
           </struct_i>
           <struct_i>
               <stringParm name="s1Ap1">Value 2</stringParm>
           </struct_i>
         </arrayOfStruct>
    </structParm>
    <arrayOfStructParm name="s2Ref" struct="s2" count="2" passDirection="in" >
      <struct_i>
          <stringParm name="s2p1">Value 1</stringParm>
      </struct_i>
      <struct_i>
          <stringParm name="s2p1">Value 2</stringParm>
      </struct_i>
    </arrayOfStructParm>
  </parameterList>
 </program>
</xpcml>
```

例如，下列 XPCML 會指定 3 intParms 的陣列，並將第一個元素設爲 12、第二個設爲 100，第三個設爲 4：

```
<?xml version="1.0"?>
<xpcml version="4.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation='xpcml.xsd' >
  <program name="prog1" path="/QSYS.lib/MYLIG.lib/PROG1.pgm">
  <parameterList>
    <arrayOfIntParm name="intArray" count="3">
      <i>12</i>
      <i>100</i>
      <i>4</i>
    </arrayOfIntParm>
  </parameterList>
  </program>
</xpcml>
```

使用 <i> 及 <struct_i> 標籤的索引屬性來設定陣列值

您可以使用 <i> 及 <struct_i> 標籤的索引屬性來協助您設定陣列值。在下列範例中，XPCML 會將第一個陣列元素設爲 4，第二個設爲 100，第三個設爲 12：

```
<?xml version="1.0"?>
<xpcml version="4.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation='xpcml.xsd' >

  <program name="prog1" path="/QSYS.lib/MYLIG.lib/PROG1.pgm">
  <parameterList>
    <arrayOfIntParm name="intArray" count="3">
      <i index="2">12</i>
      <i index="1">100</i>
      <i index="0">4</i>
    </arrayOfIntParm>
  </parameterList>
  </program>
</xpcml>
```

範例：壓縮現有的 XPCML 文件

下列範例說明如何壓縮現有的 XPCML 文件。範例中包含了原始 XPCML 來源檔、已壓縮的結果 XPCML 以及延伸的綱目。

原始 XPCML 來源

```
<?xml version="1.0" encoding="UTF-8"?>
 <xpcml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="xpcml.xsd" version="4.0">
  <program name="prog1" path="/QSYS.LIB/W95LIB.LIB/PROG1.PGM">
    <parameterList>
       <stringParm name="parm1" passDirection="in" passMode="value"
             minvrm="V5R2M0" ccsid="37" length="10">Value 1</stringParm>
    </parameterList>
  </program>
 </xpcml>
```

壓縮 XPCML 來源

```
<?xml version="1.0" encoding="UTF-8"?>
<xpcml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="myXSD.xsd" version="4.0">
 <program name="prog1" path="/QSYS.LIB/W95LIB.LIB/PROG1.PGM">
    <parameterList>
       <parm1_>Value 1</parm1_>
    </parameterList>
 </program>
</xpcml>
```

產生的綱目

```
<!-- parm1's XSD definition -->
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>
 <!-- Link back to XPCML.xsd -->
 <xs:include schemaLocation='xpcml.xsd'/>
  <xs:element name="parm1_" substitutionGroup="stringParmGroup" >
   <xs:complexType>
     <xs:simpleContent>
     <xs:restriction base="stringParmType">
        <!-- Attributes defined for parm1 -->
        <xs:attribute name="name" type="string50" fixed="parm1" />
        <xs:attribute name="length" type="xs:string" fixed="10" />
        <xs:attribute name="passMode" type="xs:string" fixed="value" />
        <xs:attribute name="ccsid" type="xs:string" fixed="37" />
        <xs:attribute name="minvrm" type="xs:string" fixed="V5R2M0" />
     </xs:restriction>
     </xs:simpleContent>
   </xs:complexType>
  </xs:element>
</schema>
```

範例：壓縮現有的 XPCML 文件，包括 Java 程式碼

下列範例說明如何壓縮現有的 XPCML 文件。範例中包含了原始 XPCML 來源檔、已壓縮的結果 XPCML、呼叫 condenseXPCML() 的 Java 程式碼，以及延伸綱目中幾個新產生的類型定義：

原始 XPCML 來源

```
<!-- Fully specified XPCML source -->
<xpcml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="xpcml.xsd" version="4.0">

<struct name="qualifiedJobName">
   <stringParm name="jobName" length="10">*</stringParm>
```

```
   <stringParm name="userName" length="10"/>
   <stringParm name="jobNumber" length="6"/>
</struct>

<struct name="jobi0100">
   <intParm name="numberOfBytesReturned"/>
   <intParm name="numberOfBytesAvailable"/>
   <structParm name="qualifiedJobName" struct="qualifiedJobName"/>
   <hexBinaryParm name="internalJobIdentifier" totalBytes="16"/>
   <stringParm name="jobStatus" length="10"/>
   <stringParm name="jobType" length="1"/>
   <stringParm name="jobSubtype" length="1"/>
   <stringParm length="2"/>
   <intParm name="runPriority"/>
   <intParm name="timeSlice"/>
   <intParm name="defaultWait"/>
   <stringParm name="purge" length="10"/>
</struct>

<program name="qusrjobi_jobi0100" path="/QSYS.LIB/QUSRJOBI.PGM">
   <parameterList>
      <structParm name="receiverVariable" passDirection="out"
         outputSize="lengthOfReceiverVariable" struct="jobi0100"/>
      <intParm name="lengthOfReceiverVariable" passDirection="in">86</intParm>
      <stringParm name="formatName" passDirection="in" length="8">JOBI0100</stringParm>
      <structParm name="qualifiedJobName" passDirection="in" struct="qualifiedJobName"/>
      <hexBinaryParm name="internalJobIdentifier"
         passDirection="in" totalBytes="16"> </hexBinaryParm>
      <intParm name="errorCode" passDirection="in">0</intParm>
   </parameterList>
</program>
</xpcml>
```

用來壓縮原始 XPCML 來源的 Java 程式碼

```
      try {
   FileInputStream fullStream = new FileInputStream("myXPCML.xpcml");
   FileOutputStream condensedStream = new FileOutputStream("myCondensedXPCML.xpcml");
   FileOutputStream xsdStream = new FileOutputStream("myXSD.xsd");
   ProgramCallDocument.condenseXPCML(fullStream, xsdStream, condensedStream, "myXSD.xsd");

}
catch (Exception e) {
  System.out.println("error:  - "+e.getMessage());
  e.printStackTrace();
}
```

壓縮的 XPCML 來源：myCondensedXPCML.xpcml

```
<xpcml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="myXSD.xsd" version="4.0">

<struct name="qualifiedJobName">
   <jobName_>*</jobName_>
   <userName_/>
   <jobNumber_/>
</struct>

<struct name="jobi0100">
   <numberOfBytesReturned_/>
   <numberOfBytesAvailable_/>
   <structParm name="qualifiedJobName" struct="qualifiedJobName"/>
   <internalJobIdentifier_/>
   <jobStatus_/>
   <jobType_/>
   <jobSubtype_/>
   <stringParm length="2"/>
```

```
   <runPriority_/>
   <timeSlice_/>
   <defaultWait_/>
   <purge_/>
</struct>

<program name="qusrjobi_jobi0100" path="/QSYS.LIB/QUSRJOBI.PGM">
   <parameterList>
      <structParm name="receiverVariable" passDirection="out"
         outputSize="lengthOfReceiverVariable" struct="jobi0100"/>
      <lengthOfReceiverVariable_>86</lengthOfReceiverVariable_>
      <formatName_>JOBI0100</formatName_>
      <structParm name="qualifiedJobName" passDirection="in" struct="qualifiedJobName"/>
      <internalJobIdentifier_> </internalJobIdentifier_>
      <errorCode_>0</errorCode_>
   </parameterList>
</program>
</xpcml>
```

產生之綱目中的一些類型定義：myXSD.xsd

```
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>
<xs:include schemaLocation='xpcml.xsd'/>

<xs:element name="jobName_" substitutionGroup="stringParmGroup" >
   <xs:complexType>
      <xs:simpleContent>
      <xs:restriction base="stringParmType">
         <xs:attribute name="name" type="string50" fixed="jobName" />
         <xs:attribute name="length" type="xs:string" fixed="10" />
      </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>

<xs:element name="userName_" substitutionGroup="stringParmGroup" >
   <xs:complexType>
      <xs:simpleContent>
      <xs:restriction base="stringParmType">
         <xs:attribute name="name" type="string50" fixed="userName" />
         <xs:attribute name="length" type="xs:string" fixed="10" />
      </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>

<xs:element name="jobNumber_" substitutionGroup="stringParmGroup" >
   <xs:complexType>
      <xs:simpleContent>
      <xs:restriction base="stringParmType">
         <xs:attribute name="name" type="string50" fixed="jobNumber" />
         <xs:attribute name="length" type="xs:string" fixed="6" />
      </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>

<xs:element name="lengthOfReceiverVariable_" substitutionGroup="intParmGroup" >
   <xs:complexType>
      <xs:simpleContent>
      <xs:restriction base="intParmType">
         <xs:attribute name="name" type="string50" fixed="lengthOfReceiverVariable" />
         <xs:attribute name="passDirection" type="passDirectionType" fixed="in" />
      </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

```
<xs:element name="formatName_" substitutionGroup="stringParmGroup" >
   <xs:complexType>
      <xs:simpleContent>
      <xs:restriction base="stringParmType">
         <xs:attribute name="name" type="string50" fixed="formatName" />
         <xs:attribute name="length" type="xs:string" fixed="8" />
         <xs:attribute name="passDirection" type="passDirectionType" fixed="in" />
      </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>

<!-- More type definitions for each newly defined type follow here -->
</xs:schema>
```

範例：使用壓縮的 XPCML 來建立 ProgramCallDocument 物件

有些 ProgramCallDocument 建構子可接受壓縮的 XPCML 來源檔以及對應的綱目 (.xsd file)。這讓您可以使用壓縮的 XPCML 來建立 ProgramCallDocument 物件。

先前提到的建構子要求您提供下列參數：

- 指定壓縮 XPCML 檔的字串
- 包含透過執行 condenseXPCML() 而建立之類型定義的 InputStream

使用這些建構子就能載入與剖析壓縮的 XPCML 檔。此外，該處理程序會記錄所有的解析錯誤。剖析完成之後，建構子就會建立 ProgramCallDocument 物件。

下列 Java 程式碼範例會使用壓縮的 XPCML 來建立 ProgramCallDocument 物件。範例程式碼的假設如下

- 壓縮的 XPCML 檔名爲 myCondensedXPCML.xpcml
- 延伸的綱目名稱爲 myXSD.xsd

接著，程式碼會使用 ProgramCallDocument 物件來執行程式 qusrjobi_jobi0100。

```
AS400 system = new AS400();
// Create a ProgramCallDocument and parse the file.
ProgramCallDocument xpcmlDoc =
  new ProgramCallDocument(system,
                          "myCondensedXPCML.xpcml",
                          new FileInputStream("myXSD.xsd"));
boolean rc = xpcmlDoc.callProgram("qusrjobi_jobi0100");
```

註: 您用來 (於建立 ProgramCallDocument 之後) 呼叫程式的 XPCML 程式碼，即爲您要用於 PCML 的程式碼。

範例：取得程式呼叫結果作為壓縮的 XPCML

您可以使用同樣的處理程序來取得程式呼叫結果，作爲壓縮的 XPCML 或非壓縮的 XPCML。您只要呼叫 ProgramCallDocument.generateXPCML() 即可完成這一工作。

使用 setXsdName() 來指定延伸之綱目的名稱，generateXPCML() 會用它來產生壓縮的 XPCML 中 <xpcml> 標籤的 noNamespaceSchemaLocation 屬性。

當您想要使用壓縮 XPCML 中的程式呼叫結果來作爲另一個 ProgramCallDocument 物件的來源檔時，就務必要使用 setXsdName()。您必須指定延伸之綱目的名稱，以便讓剖析器得知應使用哪一個綱目來進行剖析。

例如，下列程式碼會從程式呼叫中取得結果並產生壓縮的 XPCML。

```
AS400 system = new AS400();

// Create a ProgramCallDocument and parse the file.
ProgramCallDocument xpcmlDoc =
   new ProgramCallDocument(system, "myCondensedXPCML.xpcml", new FileInputStream("myXSD.xsd"));

boolean rc = xpcmlDoc.callProgram("qusrjobi_jobi0100");
if (rc)     // Program was successful
{
  xpcmlDoc.setXsdName("myXSD.xsd");
  xpcmlDoc.generateXPCML("qusrjobi_jobi0100","XPCMLOut.xpcml");
}
```

下列程式碼將說明如何取得程式呼叫結果作爲壓縮的 XPCML：

```
<xpcml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="myXSD.xsd" version="4.0">

<program name="qusrjobi_jobi0100" path="/QSYS.LIB/QUSRJOBI.PGM">
   <parameterList>
   <structParm name="receiverVariable" passDirection="out"
       outputSize="lengthOfReceiverVariable" struct="jobi0100"/>
            <numberOfBytesReturned_>100</numberOfBytesReturned_>
            <numberOfBytesAvailable_>100</numberOfBytesAvailable_>
            <structParm name="qualifiedJobName"
                struct="qualifiedJobName">
                <jobName_>*</jobName_>
                <userName_/>
                <jobNumber_/>
            </structParm>
            <internalJobIdentifier_/>
            <jobStatus_>ACTIVE</jobStatus_>
            <jobType_>PJ</jobType>
            <jobSubtype_/>
            <stringParm length="2"/>
            <runPriority_>5</runPriority>
            <timeSlice_/>
            <defaultWait_>10</defaultWait_>
            <purge_/>
        </structParm>
        <lengthOfReceiverVariable_>86</lengthOfReceiverVariable_>
        <formatName_>JOBI0100</formatName_>
        <structParm name="qualifiedJobName" passDirection="in" struct="qualifiedJobName"/>
        <internalJobIdentifier_> </internalJobIdentifier_>
        <errorCode_>0</errorCode_>
</parameterList>
</program>
</xpcml>
```

IBM Toolbox for Java 的相關資訊

下列清單中包含與 IBM Toolbox for Java 資訊相關的網站及「資訊中心」主題。

IBM Toolbox for Java 資源

請利用下列網站進一步瞭解 IBM Toolbox for Java：

- IBM Toolbox for Java and JTOpen  ：提供服務修正程式包、效能要訣、範例等等豐富的資訊。您也可以下載這份資訊的已壓縮套件，包括 javadocs 在內。
- IBM Toolbox for Java 常見問題集 (FAQ)  ：提供效能、疑難排解、JDBC 等等相關問題的回答。

- IBM Toolbox for Java and JTOpen 論壇  ：提供有效方式，以與使用 IBM Toolbox for Java 的 Java 程式設計師的社群及 IBM Toolbox for Java 開發者本人進行通訊。

IBM Toolbox for Java 2 Micro Edition 資源

請利用下列網站進一步瞭解 ToolboxME for iSeries 及 Java 實作的無線技術：

- IBM Toolbox for Java and JTOpen  ：提供 ToolboxME for iSeries 的相關資訊。
- IBM alphaWorks® Wireless  ：提供新無線技術的相關資訊，包括開發資源的下載及鏈結。
- Sun Java 2 Platform, Micro Edition  ：提供 Java 無線技術的相關資訊，包括下列各項：
  - K Virtual Machine (KVM)
  - Connected Limited Device Configuration (CLDC)
  - 行動資訊裝置設定檔 (MIDP)
- Java Wireless Developer  ：爲 Java 無線應用程式開發者提供廣泛的技術資訊。
- 無線通訊應用程式開發工具：
  - IBM WebSphere Studio Device Developer 
  - Java 2 Platform Micro Edition, Wireless Toolkit 

Java

Java 是可讓您開發可攜性物件導向應用程式及 Applet 的程式設計語言。請利用下列網站進一步瞭解 Java：

- IBM developerWorks® Java technology 區域  ：提供資訊、教育及工具，用於協助您使用 Java、IBM 產品及其他技術，以建立商務解決方案。
- IBM alphaWorks Java  ：提供 Java 最新技術的相關資訊，包括開發資源的下載及鏈結。
- Sun Microsystems 提供的 The Source for Java Technology  ：提供 Java 各種用法的相關資訊，其中也包括新技術在內。

Java Naming and Directory Interface

- Java Naming and Directory Interface(TM) (JNDI)  ：提供 JNDI 的概觀、技術資訊、範例，以及可利用的服務提供者名單。
- iSeries Directory Server (LDAP)  ：提供在 i5/OS 中使用 LDAP (Lightweight Directory Access Protocol) 的相關資訊。

Java Secure Socket Extension

- Java Secure Socket Extension (JSSE)  ：提供 JSSE 的概觀及更多資訊的鏈結。

Servlet

Servlet 是在伺服器中執行的小型 Java 程式，能夠調解一個或眾多用戶端 (各在瀏覽器中執行) 對一或多個資料庫的要求。因爲 Servlet 是使用 Java 程式設計而成，所以能夠在單一處理中以多重緒來執行要求，從而節省系統資源。請利用下列網站進一步瞭解 servlet：

- IBM Websphere, IBM PartnerWorld®  ：提供 Servlet 型 Web 應用程式伺服器的相關資訊。
- Java Servlet technology  ：提供技術資訊、指示及工具，用於協助您瞭解及運用 Servlet。

XHTML

XHTML 因爲是 HTML 4.0 的繼承者而受到重視。它根據 HTML 4.0， 但納入了 XML 的延伸性。請利用下列網站進一步瞭解 XHTML：

- The Web Developer's Virtual Library  ：提供 XHTML 的介紹，包括範例和更多資訊的鏈結。
- W3C  ：提供 XHTML 標準建議的相關技術資訊。

XML

Extensible Markup Language (XML) 是可讓您用人類和電腦都很容易瞭解的方式來說明和組織資訊的中繼語言。中繼語言可讓您定義文件標記語言和這種語言的結構。請利用下列網站進一步瞭解 XML：

- IBM developerWorks XML 區域  ：提供一個專門用於 IBM 研發 XML 工作以及它如何協助電子商務的網站
- IBM alphaWorks XML  ：提供新興 XML 標準及工具的相關資訊，包括開發資源的下載及鏈結。
- W3C XML  ：提供 XML 程式開發者的技術資源。
- XML.com  ：提供 XML 於電腦業界的最新資訊
- XML.org  ：提供 XML 社群的新聞和資訊，包括產業新聞、行事曆等等。
- XML Cover Pages  ：提供 XML、SGML 和相關的 XML 標準如 XSL、XSLT 的綜合性線上參考文章。

其它參考資料

- IBM HTTP Server for iSeries  ：提供 IBM HTTP Server for iSeries 的相關資訊、資源及要訣。
- iSeries Access for Windows  ：提供 iSeries Access for Windows 的相關資訊，包括下載、常見問題集 (FAQ)，以及其他網站的鏈結。
- IBM WebSphere Host On-Demand  ：提供瀏覽器型模擬程式的相關資訊，該模擬程式提供對 S/390®、iSeries 及 DEC/Unix 模擬的支援。
- IBM Support and downloads  ：提供 IBM 軟硬體支援的入口網站。

程式碼授權及免責聲明資訊

IBM 授與您使用所有程式設計程式碼範例的非專屬授權，您可以利用這些範例來產生符合您需求的類似函數。

除法律規定不得排除的保證外，IBM、IBM 之程式開發人員及供應商不附具任何明示或默示之保證，包含且不限於任何相關技術支援之未侵害他人權利之保證、或可商用性及符合特定效用等之默示保證。

在任何情況下，IBM、IBM 之程式開發者或供應商對下列情事均不負賠償責任，即使被告知該情事有可能發生時，亦同：

1. 資料之滅失或毀損；
2. 直接、特殊、附帶或間接的傷害或其他衍生之經濟損害；或
3. 利潤、營業、收益、商譽或預期節餘等項之損失。

倘法律規定不得排除或限制賠償責任時，則該排除或限制無效。

條款

根據下述條款，授予您對這些出版品的使用權限。

個人使用：您可複製該等出版品供個人及非商業性用途使用，惟應註記 IBM 著作權標示及其他所有權歸屬 IBM 之相關文字。未經 IBM 明示同意，您不得散佈、展示或改作該等出版品或其任何部分。

商業使用：您可以複製、散佈及展示該等出版品僅供企業內部使用，惟應註記 IBM 著作權標示及其他所有權歸屬 IBM 之相關文字。未經 IBM 明示同意，您不得改作該等出版品，也不得於企業外複製、散佈或展示該等出版品或其任何部分。

除本使用聲明中明確授予之許可外，使用者就出版品或任何包含於其中之資訊、資料、軟體或其他智慧財產權，並未取得其他任何明示或默許之許可、軟體授權或權利。

使用者對於出版品之使用如危害 IBM 的權益，或 IBM 認定其未遵照上述指示使用出版品時，IBM 得隨時撤銷此處所授予之許可。

除非您完全遵守所有適用之一切法規，包括所有美國出口法規，否則您不得下載、出口或再輸出此等資訊。

IBM 對於該等出版品之內容不爲任何保證。出版品依其「現狀」提供，不附帶任何明示或默示之擔保，其中包括 (但不限於) 適售性、未涉侵權及適合特定用途之默示擔保責任。

附錄. 注意事項

本資訊是針對 IBM 在美國所提供之產品與服務開發出來的。

而在其他國家中，IBM 不見得有提供本書中所提的各項產品、服務、或功能。要知道您所在區域是否可用到這些產品與服務時，請向當地的 IBM 服務代表查詢。本書在提及 IBM 產品、程式或服務時，不表示或暗示只能使用 IBM 的產品、程式或服務。只要未侵犯 IBM 的智慧財產權，任何功能相當的產品、程式或服務都可以取代 IBM 的產品、程式或服務。不過，其他非 IBM 產品、程式或服務在運作上的評價與驗證，其責任屬於使用者。

在這本書或文件中可能包含著 IBM 所擁有之專利或專利申請案。本書使用者並不享有前述專利之任何授權。您可以用書面方式來查詢授權，來函請寄到：

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

若要查詢有關二位元組 (DBCS) 資訊的特許權限事宜，請聯絡您國家的 IBM 智慧財產部門，或者用書面方式寄到：

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

下列段落若與當地之法令抵觸，則不適用之： IBM 僅以「現狀」提供本出版品，而不爲任何明示或默示之保證 (包括但不限於產品未涉侵權、可售性或符合特定效用的保證。) 倘若干地區在特定交易中並不許可相關明示或默示保證之棄權聲明，則於該等地區之特定交易，此項聲明不適用之。

本資訊中可能包含技術上或排版印刷上的錯誤。因此，IBM 會定期修訂；並將修訂後的內容納入新版中。同時，IBM 得隨時修改或變更本出版品中所提及的產品及程式。

本資訊中任何對非 IBM 網站的敘述僅供參考，IBM 對該等網站並不提供保證。該等網站上的資料，並非 IBM 產品所用資料的一部分，如因使用該等網站而造成損害，其責任由 貴客戶自行負責。

IBM 得以其認定之各種適當方式使用或散布由 貴客戶提供的任何資訊，而無需對您負責。

本程式之獲授權者若希望取得相關資料，以便使用下列資訊者可洽詢 IBM。其下列資訊指的是： (1) 獨立建立的程式與其他程式 (包括此程式) 之間更換資訊的方式 (2) 相互使用已交換之資訊方法 若有任何問題請聯絡：

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

上述資料之取得有其特殊要件，在某些情況下必須付費方得使用。

IBM 基於雙方之「IBM 客戶合約」、「IBM 國際程式授權合約」、「IBM 機器碼授權合約」或任何同等合約之條款，提供本出版品中所述之授權程式與其所有適用的授權資料。

任何此處涵蓋的執行效能資料都是在一個受控制的環境下決定出來的。因此，於其他不同作業環境之下所得的結果，可能會有很大的差異。有些測定已在開發階段系統上做過，不過這並不保證在一般系統上會出現相同結果。再者，有些測定可能已透過推測方式評估過。但實際結果可能並非如此。本文件的使用者應根據其特有的環境，驗證出適用的資料。

本資訊所提及之非 IBM 產品資訊，係一由產品的供應商，或其出版的聲明或其他公開管道取得。IBM 並未測試過這些產品，也無法確認這些非 IBM 產品的執行效能、相容性、或任何對產品的其他主張是否完全無誤。如果您對非 IBM 產品的性能有任何的疑問，請逕向該產品的供應商查詢。

有關 IBM 未來動向的任何陳述，僅代表 IBM 的目標而已，並可能於未事先聲明的情況下有所變動或撤回。

所有顯示之 IBM 產品售價僅爲 IBM 產品之一般市場價格，可能於未事先聲明之情況下有所變動。經銷商售價可能有所不同。

本資訊僅供規劃用途。所提及的產品發行之前，本書內含的資訊有變動的可能。

本資訊中含有日常商業活動所用的資料及報告範例。爲了提供完整的說明，這些範例包括個人、公司、廠牌和產品的名稱。這些名稱全屬虛構，若與任何公司的名稱和住址雷同，純屬巧合。

著作權授權：

本資訊包含原始語言的範例應用程式，用以說明各種作業平台上的程式設計技術。您可以基於研發、使用、銷售或散佈符合作業平台 (用於執行所撰寫的範例程式) 之應用程式設計介面的應用程式等目的，以任何形式複製、修改及散佈這些範例程式，而無需付費給 IBM。這些範例尚未徹底經過所有情況的測試。因此，IBM 不保證或暗示這些程式的穩定性、服務能力或功能。

這些範例程式或是任何衍生著作的每一份拷貝或任何部分，都必須具有下列的著作權聲明：

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

若您是以電子檔檢視此資訊，則照片和彩色圖例可能不會出現。

程式設計介面資訊

本 IBM Toolbox for Java 出版品文件是使用允許客戶撰寫程式以取得 IBM Toolbox for Java 服務的「程式設計介面」。

商標

下列術語是 IBM 公司在美國及 (或) 其它國家的商標。

- Advanced Function Presentation
- Advanced Function Printing
- AFP
- AIX
- alphaWorks
- AS/400
- DB2

| DB2 Universal Database
| developerWorks
| i5/OS
| IBM
| IPDS
| iSeries
| NetServer
| OS/400
| PartnerWorld
| S/390
| SecureWay
| VisualAge
| WebSphere

Microsoft、Windows、Windows NT 以及 Windows 商標是 Microsoft Corporation 在美國及 (或) 其它國家的商標。

Java 以及所有與 Java 有關的商標是 Sun Microsystems, Inc. 在美國及 (或) 其它國家的商標。

| Linux 是 Linus TorvaIds 在美國及 (或) 其他國家的商標。

UNIX 是 The Open Group 在美國及其它國家的註冊商標。

其他公司、產品及服務名稱，可能是其他公司的商標或服務標誌。

條款

根據下述條款，授予您對這些出版品的使用權限。

個人使用：您可複製該等出版品供個人及非商業性用途使用，惟應註記 IBM 著作權標示及其他所有權歸屬 IBM 之相關文字。未經 IBM 明示同意，您不得散佈、展示或改作該等出版品或其任何部分。

商業使用：您可以複製、散佈及展示該等出版品僅供企業內部使用，惟應註記 IBM 著作權標示及其他所有權歸屬 IBM 之相關文字。未經 IBM 明示同意，您不得改作該等出版品，也不得於企業外複製、散佈或展示該等出版品或其任何部分。

除本使用聲明中明確授予之許可外，使用者就出版品或任何包含於其中之資訊、資料、軟體或其他智慧財產權，並未取得其他任何明示或默許之許可、軟體授權或權利。

使用者對於出版品之使用如危害 IBM 的權益，或 IBM 認定其未遵照上述指示使用出版品時，IBM 得隨時撤銷此處所授予之許可。

除非您完全遵守所有適用之一切法規，包括所有美國出口法規，否則您不得下載、出口或再輸出此等資訊。

IBM 對於該等出版品之內容不爲任何保證。出版品依其「現狀」提供，不附帶任何明示或默示之擔保，其中包括 (但不限於) 適售性、未涉侵權及適合特定用途之默示擔保責任。

讀者意見表

為使本書盡善盡美，本公司極需您寶貴的意見；懇請您閱讀後，撥冗填寫下表，惠予指教。

請於下表適當空格內，填入記號 (∨)；我們會在下一版中，作適當修訂，謝謝您的合作!

評估項目	評 估 意 見		備 註
正 確 性	內容說明與實際程序是否符合	□是□否	
	參考書目是否正確	□是□否	
一 致 性	文句用語及風格，前後是否一致	□是□否	
	實際產品介面訊息與本書中所提是否一致	□是□否	
完 整 性	是否遺漏您想知道的項目	□是□否	
	字句、章節是否有遺漏	□是□否	
術語使用	術語之使用是否恰當	□是□否	
	術語之使用，前後是否一致	□是□否	
可 讀 性	文句用語是否通順	□是□否	
	有否不知所云之處	□是□否	
內容說明	內容說明是否詳盡	□是□否	
	例題說明是否詳盡	□是□否	
排版方式	本書的形狀大小，版面安排是否方便閱讀	□是□否	
	字體大小，顏色編排，是否有助於閱讀	□是□否	
目錄索引	目錄內容之編排，是否便於查找	□是□否	
	索引語錄之排定，是否便於查找	□是□否	
	※評估意見為 "否" 者，請於備註欄提供建議。		

其他： (篇幅不夠時，請另外附紙說明。)

上述改正意見，一經採用，本公司有合法之使用及發佈權利，特此聲明。

註： 您也可將寶貴的意見以電子郵件寄至 tscadmin@tw.ibm.com，謝謝。

IBM 系統 - iSeries

RZAH-H000-09

程式設計
IBM Toolbox for Java

版本 5 版次 4